ALSTOM

ACCUTRACK PRODUCT SOLUTIONS

Integrated Vital Processor Interlocking (iVPI®)

Computer Aided Application (CAA)

Reference Manual

Copyright © 2009, 2011, 2013, 2014, 2015 Alstom Signaling Inc.

⚠WARNING

READ MANUAL BEFORE USE

Read this manual before using this software.

Keep this manual in a safe location for future reference.

Failure to follow the instructions presented in this manual can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

Reference Manual
P2512F

ALSTOM

ACCUTRACK PRODUCT SOLUTIONS

Integrated Vital Processor Interlocking (iVPI®)

Computer Aided Application (CAA)

Reference Manual

Copyright © 2009, 2011, 2013, 2014, 2015 Alstom Signaling Inc.

⚠WARNING

READ MANUAL BEFORE USE

Read this manual before using this software.

Keep this manual in a safe location for future reference.

Failure to follow the instructions presented in this manual can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

Reference Manual
Alstom Signaling Inc.

P2512F, Rev. G, August 2015, Printed in U.S.A.

LIST OF EFFECTIVE PAGES

P2512F, Integrated Vital Processor Interlocking (iVPI®) Computer Aided Application (CAA) Reference Manual

ORIGINAL ISSUE DATE:	October 2009
CURRENT REVISION AND DATE:	Rev G, August 2015
PAGE	CHANGE OR REVISION LEVEL
Cover	Aug/15
Title page	Aug/15
Preface	Aug/15
i through xiv	Aug/15
1-1 through 1-16	Aug/15
2-1 through 2-6	Aug/15
3-1 through 3-2	Aug/15
4-1 through 4-6	Aug/15
5-1 through 5-116	Aug/15
6-1 through 6-6	Aug/15
7-1 through 7-18	Aug/15
8-1 through 8-2	Aug/15
9-1 through 9-42	Aug/15
10-1 through 10-14	Aug/15
11-1 through 11-58	Aug/15
12-1 through 12-54	Aug/15

13-1 through 13-12	Aug/15
14-1 through 14-2	Aug/15
A-1 through A-54	Aug/15
B-1 through B-8	Aug/15

PREFACE

NOTICE OF CONFIDENTIAL INFORMATION

Information contained herein is confidential and is the property of Alstom Signaling Inc. Where furnished with a proposal, the recipient shall use it solely to evaluate the proposal. Where furnished to a customer, it shall be used solely for the purposes of inspection, installation, or maintenance. Where furnished to a supplier, it shall be used solely in the performance of the contract. The information shall not be used or disclosed by the recipient for any other purposes whatsoever.

VPI® and iVPI® are registered trademarks of Alstom Signaling Inc. All other trademarks referenced herein are trademarks of their respective owners.

FOR QUESTIONS AND INQUIRIES, CONTACT CUSTOMER SERVICE

Address:	Alstom Signaling Inc. 1025 John Street West Henrietta, NY 14586 USA
Website:	www.alstomsignalingsolutions.com
Email:	websiteinfo@alstomsignalingsolutions.com
Phone:	1-800-717-4477

REVISION LOG

Revision	Date	Description	Author	Checker	Approver
1 (A)	October 2009	Original Release	MAS	JM	NI
2 (B)	February 2011	Updated to support board changes	MAS	KW	NI
3 (C)	January 2013	Updated to add new iVPI detail	SG	KW	NI
D	November 2013	Updated to clarify details and incorporate new ADV checklist	SG	KW	MS
E	June 2014	Revised Appendix A	SG	KW	MS
F	June 2015	Revised warnings, added new Section 1, updated ADV checklist	SG	KW	MS
G	August 2015	Updated to incorporate Alternate Flashing details; added Appendix B	SG	KW	MS

ABOUT THIS MANUAL

This manual provides the reference information needed for the iVPI Computer Aided Application (CAA) portion of Computer-Aided Application Programming Environment software (CAAPE). The information in this manual is arranged into sections. The title and a brief description of each section follow:

Section 1 – SAFETY PRECAUTIONS: This section contains safety precautions applicable to the iVPI CAA.

Section 2 – GENERAL: This section provides general information on manual intent, content, and conventions.

Section 3 – INTRODUCTION: This section introduces iVPI CAA and discusses its theory of operation.

Section 4 – GENERAL RULES AND NOMENCLATURE: This section describes the basic rules for the format of application data.

Section 5 – APPLICATION RULES: This section describes aspects of iVPI hardware rules and operation that have a bearing on iVPI CAA input.

Section 6 – INPUT FILE ORGANIZATION: This section describes general rules and recommendations for organizing input files.

Section 7 – HARDWARE FILE: This section describes the hardware description file used in Vital and non-vital applications.

Section 8 – VSP-NVSP COMMUNICATION FILE: This section describes the VSP Communications file used to describe communications between Vital and non-vital applications.

Section 9 – COMPILER FILES: This section describes the files specific to the iVPI (Vital) compiler.

Section 10 – COMM COMPILER FILES: This section describes the files specific to the iVPI COMM (Vital Comm) compiler.

Section 11 – NVSP COMPILER FILES: This section describes the files specific to the NVSP (non-vital) compiler.

Section 12 – APPLICATION DATA VERIFIER PROGRAM: This section describes the ADV program and its reports.

Section 13 – ADV CONSOLIDATION REPORTS: This section lists expected consolidation report results.

Section 14 – ADV COMPARE PROGRAM: This section describes the ADV Compare program and its reports.

Appendix A – APPLICATION DATA VERIFICATION DATA SHEET: This section contains the data sheet checklist to record all necessary process steps required to validate information contained in the iVPI application before beginning revenue service.

Appendix B – ADV COMPARE CHECKLIST: This section contains the data sheet checklist to record all necessary process steps required to validate information that has changed between two applications.

MANUAL SPECIAL NOTATIONS

In Alstom manuals, signal word panels are used to convey special safety and informational messages. *Danger*, *Warning* and *Caution* Signal Word Panels include a safety alert symbol in addition to the signal word, whereas *Notice* and *Safety Instructions* Signal Word Panels include only the signal word.

This is the safety alert symbol. It is used to alert of potential physical injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

Signal Word Panel	Definition
DANGER	Indicates a hazardous situation, which, if not avoided, **will** result in death or serious injury.
WARNING	Indicates a hazardous situation, which, if not avoided, **could** result in death or serious injury.
CAUTION	Indicates a hazardous situation, which, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a situation which could result in property damage or information considered important, but not hazard-related.
SAFETY INSTRUCTIONS	Indicates specific safety-related instructions or procedures.

TABLE OF CONTENTS

Topic	Page
SECTION 1 – Safety Precautions	**1-1**
1.1 Safety Precaution Matrix	1-1
1.2 Safety Precautions	1-3
SECTION 2 – General	**2-1**
2.1 Safety	2-1
2.2 Intended Audience	2-2
2.3 Document Conventions	2-2
2.4 Common Abbreviations and Glossary	2-3
2.5 Related Publications	2-5
SECTION 3 – Introduction	**3-1**
3.1 Theory of Operation	3-2
SECTION 4 – General Rules and Nomenclature	**4-1**
4.1 Records	4-1
4.2 Default Values	4-1
4.3 Free Form Data	4-1
4.4 Comment Records	4-2
4.5 Drawing Numbers	4-3
4.6 Parameter Names	4-4
4.6.1 System Reserved Names	*4-5*
4.7 Board Identifier Names	4-6
SECTION 5 – iVPI Application Rules	**5-1**
5.1 Hardware Application Rules	5-2
5.1.1 Output Protection	*5-3*
5.1.2 Board Placement	*5-4*
5.1.2.1 Board Placement Rules	*5-5*
5.1.2.2 Board Address Assignment	*5-6*
5.1.3 Network Boards	*5-6*
5.1.4 Vital Input Boards	*5-7*
5.1.5 Vital Output Boards	*5-10*
5.1.5.1 Output PROMs	*5-10*
5.1.5.2 Lamp Drive Output Board	*5-13*
5.1.5.3 Single Break Output Board	*5-23*
5.1.5.4 Double Break Output Board	*5-28*

TABLE OF CONTENTS

Topic **Page**

5.1.5.5 AC Output Board 5-30
5.1.5.6 Restrictive Flashing Aspects 5-33
5.1.6 NVSP Boards 5-34
5.1.6.1 Software Revision / Site ID 5-34
5.1.7 Genrakode Track Processor (GTP) Boards 5-34
5.1.8 Code Rate Generator (CRG) Boards 5-35
5.1.9 Non-Vital Input (NVI) Boards 5-37
5.1.10 Non-Vital Output (NVO) Boards 5-38
5.2 VSP-NVSP Communications 5-40
5.2.1 VSP-to-NVSP Messages 5-40
5.2.2 NVSP-to-VSP Messages 5-41
5.2.3 NVSP-to-VSP Diagnostic Messages 5-41
5.3 GTP Communications 5-42
5.3.1 VSP-to-GTP Messages. 5-42
5.3.2 GTP-to-VSP Messages. 5-42
5.3.3 GTP Diagnostic Messages 5-43
5.3.3.1 GTP Links and Blocks 5-43
5.4 CRG Communications 5-44
5.4.1 CRG Data Messages 5-44
5.4.2 CRG Status Messages 5-44
5.4.3 CRG Diagnostic Messages. 5-45
5.4.3.1 CRG Links and Blocks 5-45
5.5 Networking 5-46
5.5.1 IP Addresses and Subnets 5-47
5.5.2 Redundancy 5-48
5.5.3 Client / Server 5-48
5.5.4 Ethernet Ports 5-49
5.6 Network Communications (VSP) 5-50
5.6.1 Introduction 5-50
5.6.2 Network-Based Diagnostics 5-51
5.6.3 Vital Serial Over Ethernet 5-52
5.6.3.1 VSOE Node Types. 5-52
5.6.3.2 Output VSOE Messages 5-52
5.6.3.3 Input VSOE Messages. 5-52
5.6.3.4 VSOE LINKOK 5-52

TABLE OF CONTENTS

Topic **Page**

5.6.3.5 IP Addresses and Network Ports 5-53
5.6.3.6 Link and Block Numbers 5-54
5.6.3.7 Coordination of VSOE Input Files 5-55
5.6.4 DigiSAFE 5-58
5.6.4.1 DigiSAFE Node Types 5-58
5.6.4.2 Output DigiSAFE Messages 5-58
5.6.4.3 Input DigiSAFE Messages 5-58
5.6.4.4 IP Addresses and Network Ports 5-59
5.6.4.5 Coordination of DigiSAFE Input Files. 5-60
5.7 Network Communications (NVSP) 5-64
5.7.1 Introduction 5-64
5.7.2 Network-Based Diagnostics 5-65
5.7.3 Network Serial Ports (NVSOE) 5-66
5.7.3.1 Serial Protocols and LPC Files 5-66
5.7.3.2 IP Addresses and Network Ports 5-67
5.7.3.3 Online Control Variable 5-68
5.7.3.4 Controls and Indications. 5-68
5.7.3.5 Unlatched Controls. 5-69
5.7.3.6 Text Messages. 5-69
5.7.3.7 Special Messages 5-69
5.7.3.8 Stations and Message Order 5-69
5.7.3.9 MAC TCP Panel Port 5-69
5.7.3.10 Coordination of NVSOE Input Files 5-70
5.8 Vital Serial Links and Blocks. 5-72
5.8.1 Link Numbers 5-73
5.8.2 Block Numbers 5-74
5.9 Non-Vital Serial Communications 5-77
5.9.1 Serial Protocols and LPC Files 5-77
5.9.2 Baud Rate and Data Format. 5-77
5.9.3 Online Control Variable. 5-78
5.9.4 Controls and Indications 5-78
5.9.5 Unlatched Controls 5-78
5.9.6 Text Messages 5-79
5.9.7 Special Messages. 5-79
5.9.8 Stations and Message Order 5-79

TABLE OF CONTENTS

Topic **Page**

5.10 Data Logging 5-80
5.10.1 Data Protection and Auto Dump 5-80
5.10.2 System Snapshots 5-80
5.10.3 Types of Log Data. 5-81
5.10.4 Log Data Output Options 5-82
5.10.5 Data Logger Names 5-82
5.11 Vital Logic 5-83
5.11.1 Internal Variables 5-83
5.11.1.1 Current Results 5-83
5.11.1.2 Self-Latched Parameters 5-83
5.11.1.3 Timer and Timer Programmable Parameters. 5-83
5.11.2 APPLICATION Statements 5-84
5.11.3 LOCATION Statements 5-84
5.11.4 Boolean Equation Statements 5-85
5.11.4.1 Logic Expression 5-85
5.11.4.2 Result List 5-86
5.11.4.3 Boolean Equations as Relay Circuits 5-86
5.11.4.4 SLOW Equations 5-86
5.11.4.5 Fixed Interval Timer Equations 5-87
5.11.4.6 Field-Settable Software Vital Timer Equations. 5-88
5.11.4.7 Using Library Files 5-89
5.11.4.8 Identifying Library Files 5-89
5.11.4.9 Inserting Library Members 5-90
5.12 Non-Vital Logic 5-91
5.12.1 Constant Declarations. 5-91
5.12.2 Internal Variables 5-92
5.12.2.1 Boolean Parameters 5-92
5.12.2.2 Integer Parameters 5-92
5.12.2.3 Timer Parameters 5-92
5.12.3 Non-Vital Logic Features 5-92
5.12.4 Arrays 5-93
5.12.5 Application Logic Execution Errors 5-93
5.12.6 Integer Arithmetic 5-94
5.12.6.1 Integer Variables 5-94
5.12.6.2 Integer Constants 5-94

TABLE OF CONTENTS

Topic **Page**

5.12.6.3 Integer Operators 5-95
5.12.6.4 Integer Equation Statements 5-95
5.12.7 Boolean Logic 5-96
5.12.7.1 Boolean Variables 5-96
5.12.7.2 Boolean Constants 5-96
5.12.7.3 Boolean Operators 5-96
5.12.7.4 Boolean Equation Statements 5-97
5.12.7.5 TIME DELAY Statements 5-98
5.12.7.6 APPLICATION Statements 5-98
5.12.8 Program Flow Control 5-99
5.12.8.1 Conditional Logic 5-99
5.12.8.2 Logical Expressions 5-99
5.12.8.3 IF..ELSE Statements 5-100
5.12.8.4 WHILE Statements 5-102
5.12.8.5 Statement Labels 5-103
5.12.8.6 GOTO Statements 5-103
5.12.8.7 SUBROUTINE Definitions 5-104
5.12.8.8 CALL Statements 5-105
5.12.9 Predefined Subroutines 5-108
5.12.9.1 Predefined Timer Subroutines 5-108
5.12.9.2 Predefined Data Input/Output Subroutines 5-110
5.12.9.3 Predefined Data Conversion Subroutines 5-112
5.12.9.4 Predefined Date/Time Subroutines 5-113
5.12.10 Predefined Variables 5-113
5.12.10.1 DPRAM_READ 5-114
5.12.10.2 DPRAM_WRITE 5-115
5.12.10.3 DPRAM_LATCH 5-116
5.12.11 Using Library Files 5-116

SECTION 6 – Input File Organization 6-1
6.1 Input Data Sections 6-1
6.2 Module Sections 6-1
6.3 Main / Include Files 6-2
6.4 System Compiler Input Files 6-3
6.5 System COMM Compiler Input Files 6-4

TABLE OF CONTENTS

Topic	Page
6.6 NVSP Compiler Input Files	6-5
6.7 Shareable Network Files	6-6
SECTION 7 – Hardware File (.HDW)	**7-1**
7.1 Basic File Format	7-1
7.2 Hardware Properties Records	7-2
7.2.1 FLASH Records	*7-2*
7.2.2 PULSE, PULSE2 Records	*7-2*
7.3 Module Properties Records	7-3
7.3.1 MODULE TYPE Records	*7-3*
7.3.2 MODULE Records	*7-3*
7.3.3 RACK Records	*7-4*
7.3.4 POSITION Records	*7-4*
7.3.5 MODULE_PREFIX Records	*7-4*
7.4 Board Assignment SUbSection	7-5
7.4.1 Board Assignment Records	*7-5*
7.4.1.1 SLOT Records	*7-5*
7.4.1.2 IONAME_n_PREFIX / IONAME_n_SUFFIX Records	*7-6*
7.4.1.3 GROUP Records	*7-7*
7.4.1.4 WIRENAME GROUP Records	*7-8*
7.4.1.5 Diagnostics Records	*7-8*
7.4.1.6 External Power Supply Records	*7-9*
7.4.1.7 WIRENAME_n_PS Records	*7-9*
7.4.1.8 I/O Port Records	*7-10*
7.4.1.9 I/O Port Mode Records	*7-10*
7.4.2 Board Type Specific Records	*7-11*
7.4.2.1 ACO Board	*7-11*
7.4.2.2 BEX Board	*7-11*
7.4.2.3 Code Rate Generator (CRG) Board	*7-12*
7.4.2.4 Genrakode Track Processor (GTP) Board	*7-12*
7.4.2.5 DBO Board	*7-13*
7.4.2.6 DI Board	*7-13*
7.4.2.7 LDO Boards	*7-14*
7.4.2.8 Non-Vital Input Boards	*7-15*
7.4.2.9 Non-Vital Output Boards	*7-16*

TABLE OF CONTENTS

Topic **Page**

7.4.2.10 NVSP Board 7-17
7.4.2.11 SBO Board 7-18
7.4.2.12 VSP Board 7-18

SECTION 8 – VSP-NVSP Communications File (.VCn) 8-1

SECTION 9 – Compiler Files 9-1
9.1 Main Compiler File (.VPC) 9-1
9.1.1 Revision History 9-1
9.1.2 File Records 9-2
9.1.2.1 COMPILER RUN CONTROLS Records 9-2
9.1.2.2 APPLICATION PROGRAM NUMBER Records 9-5
9.1.2.3 VSP PROGRAM NUMBER Records 9-6
9.1.2.4 COPYRIGHT YEAR Records 9-6
9.1.2.5 SYSTEM SOFTWARE Records 9-7
9.1.2.6 CONTRACT NUMBER Record 9-7
9.1.2.7 CONTRACT NAME Records 9-8
9.1.2.8 CUSTOMER NAME Record 9-8
9.1.2.9 EQUIPMENT LOCATION Record 9-8
9.1.2.10 DESIGNER Records 9-9
9.1.2.11 CHECKER Records 9-9
9.1.2.12 VSP ID Records 9-9
9.1.2.13 VITAL OUTPUT FLASHING Records 9-10
9.1.2.14 RESTRICTIVE FLASHING ASPECTS Records 9-11
9.1.2.15 SOFTWARE REVISION / SITE ID Records 9-12
9.1.2.16 SYSTEM ID Records 9-15
9.1.2.17 NVSP ID Records 9-16
9.1.2.18 NVSP PROGRAM NUMBER Records 9-17
9.1.2.19 GTP ID Records 9-17
9.1.2.20 GTP EXPORT FILE Records 9-18
9.1.2.21 CRG ID Records 9-18
9.1.2.22 CRG SOFTWARE Records 9-19
9.1.2.23 PASSWORD Records 9-19
9.1.2.24 iVPI ID Records 9-20
9.1.2.25 LIBRARY PATH Records 9-20
9.1.2.26 INCLUDE Records 9-21

TABLE OF CONTENTS

Topic **Page**

9.2 VSOE and DigiSAFE Nodes Declaration file (.vnt) 9-22
9.3 Vital Serial Link Definition File (.cw) 9-24
9.4 VSOE / GTP / CRG / DigiSAFE Messages File (.VSL) 9-29
9.4.1 Network VSC Section 9-29
9.4.2 DigiSAFE Section 9-30
9.4.3 GTP Communications Section 9-31
9.4.4 CRG Communications Section 9-32
9.4.5 Example 9-34
9.5 Vital Logic Files (.PRM/.VTL) 9-35
9.5.1 Current Result Parameter Declarations 9-35
9.5.2 Self-Latched Parameter Declarations 9-36
9.5.3 Timer Parameter Declarations 9-36
9.5.4 Boolean Equation Section 9-37
9.5.4.1 APPLICATION Statements 9-37
9.5.4.2 LOCATION Statements 9-38
9.5.4.3 Boolean Equation Statements 9-38
9.5.4.4 Time Delay and Time Delay Programmable Statements . . . 9-40
9.5.4.5 LIBRARY FILE Records 9-42
9.5.4.6 LIBR Records 9-42

SECTION 10 – Comm Compiler Files 10-1
10.1 Main Compiler File (.VCC) 10-1
10.1.1 Revision History 10-1
10.1.2 File Records 10-2
10.1.2.1 APPLICATION PROGRAM NUMBER Record 10-2
10.1.2.2 VSPCP PROGRAM NUMBER Record 10-3
10.1.2.3 COPYRIGHT YEAR Records 10-3
10.1.2.4 SYSTEM SOFTWARE Records 10-4
10.1.2.5 CONTRACT NUMBER Records 10-4
10.1.2.6 CONTRACT NAME Records 10-5
10.1.2.7 CUSTOMER NAME Records 10-5
10.1.2.8 EQUIPMENT LOCATION Records 10-5
10.1.2.9 DESIGNER Records 10-6
10.1.2.10 CHECKER Records 10-6
10.1.2.11 VSP ID Records 10-6

TABLE OF CONTENTS

Topic **Page**

10.1.2.12 ENET1, ENET2 Device Records 10-7
10.1.2.13 MACTCP Records 10-8
10.1.2.14 INCLUDE Record 10-8
10.2 VSOE and DigiSAFE Connection Data File (.NVS) 10-9
10.3 Gateways File (.GW) 10-13
10.4 MMS Connection Data File (.NMM) 10-14

SECTION 11 – NVSP Compiler Files 11-1
11.1 Main Compiler File (.CSI) 11-1
11.1.1 Revision History 11-1
11.1.2 File Records 11-2
11.1.2.1 COMPILER RUN CONTROLS Records 11-2
11.1.2.2 APPLICATION PROGRAM NUMBER Records 11-4
11.1.2.3 NVSP PROGRAM NUMBER Records 11-5
11.1.2.4 COPYRIGHT YEAR Record 11-5
11.1.2.5 CONTRACT NUMBER Records 11-6
11.1.2.6 CONTRACT NAME Record 11-6
11.1.2.7 CUSTOMER NAME Records 11-6
11.1.2.8 EQUIPMENT LOCATION Records 11-7
11.1.2.9 DESIGNER Record 11-7
11.1.2.10 CHECKER Records 11-7
11.1.2.11 NVSP ID Record 11-8
11.1.2.12 NVSP SYSTEM SOFTWARE Records 11-8
11.1.2.13 DATA LOGGING Records 11-9
11.1.2.14 DIAGNOSTIC PASSWORD RECORD Records 11-9
11.1.2.15 DIAGNOSTIC TERMINAL TYPE Records 11-10
11.1.2.16 DIAGNOSTIC TERMINAL BAUD RATE Records 11-10
11.1.2.17 DIAGNOSTIC TERMINAL DATA FORMAT Records 11-11
11.1.2.18 LIBRARY PATH Records 11-11
11.1.2.19 ENET1, ENET2 Device Records 11-12
11.1.2.20 MACTCP Records 11-13
11.1.2.21 VSP SOFTWARE SITE ID / VSP SYSTEM ID Records . . 11-14
11.1.2.22 SOFTWARE REVISION ID Records 11-15
11.1.2.23 INCLUDE Records 11-15
11.2 Serial Communications File (.CSS) 11-16

TABLE OF CONTENTS

Topic	Page
11.2.1 Serial Port Definition Section	*11-17*
11.2.2 Port Settings Records	*11-18*
11.2.2.1 OPERATING MODE Records	*11-18*
11.2.2.2 CONFIGURATION FILE Records	*11-18*
11.2.2.3 DEFAULT BAUD RATE Records	*11-19*
11.2.2.4 BAUD RATE CONTROL Records	*11-19*
11.2.2.5 DATA FORMAT Records	*11-20*
11.2.2.6 ONLINE CONTROL Records	*11-20*
11.2.3 Serial Messages	*11-21*
11.2.3.1 Control (Incoming) Messages	*11-22*
11.2.3.2 Indication (Outgoing) Messages	*11-23*
11.2.3.3 Text Messages	*11-24*
11.2.3.4 Special Messages	*11-25*
11.3 Network Serial Communications File (.NSS)	11-27
11.3.1 Network Port Settings Records	*11-30*
11.3.1.1 CONFIGURATION FILE Record	*11-30*
11.3.1.2 ONLINE CONTROL Record	*11-30*
11.3.1.3 REMOTE NETWORK CONNECTION Record	*11-31*
11.3.2 Messages	*11-32*
11.4 NVSOE Connection Data File (.NNS)	11-33
11.5 NVSOE Links Definition File (.NCW)	11-36
11.6 Data Logger File (.LOG)	11-39
11.6.0.1 DATA PROTECT Records	*11-40*
11.6.0.2 AUTO DUMP Record	*11-41*
11.6.0.3 SYSTEM SNAPSHOT PERIOD Record	*11-42*
11.6.0.4 DATA LOG Records	*11-43*
11.6.0.5 INPUT LOG Record	*11-44*
11.6.0.6 OUTPUT LOG Record	*11-45*
11.6.0.7 PRINT MODE Record	*11-45*
11.6.0.8 DATA LOGGING INTERFACE Record	*11-46*
11.6.0.9 DATALOGGER NAMES Record	*11-46*
11.6.1 Application Log Messages	*11-47*
11.6.2 Example	*11-47*
11.7 Non-Vital Logic Files (.PRM/.NV)	11-48
11.7.1 Constant Declarations	*11-49*

TABLE OF CONTENTS

Topic **Page**

11.7.2 Boolean Parameter Declarations 11-49
11.7.3 Integer Parameter Declarations 11-50
11.7.4 Timer Parameter Declarations 11-50
11.7.5 Non-Vital Logic Section. 11-51
11.7.5.1 Non-Vital Logic Section / Boolean Equation Section Records 11-51
11.7.5.2 Integer Equation Statements 11-51
11.7.5.3 Boolean Equation Statements 11-52
11.7.5.4 TIME DELAY Statements. 11-52
11.7.5.5 APPLICATION Statements 11-53
11.7.5.6 IF..ELSE Statement 11-53
11.7.5.7 WHILE Statements. 11-55
11.7.5.8 Statement Labels 11-55
11.7.5.9 GOTO Statements 11-56
11.7.5.10 SUBROUTINE Definitions 11-56
11.7.5.11 CALL Statements 11-57
11.7.5.12 LIBRARY FILE Records. 11-57
11.7.5.13 LIBR Records. 11-58

SECTION 12 – Application Data Verifier Program 12-1
12.1 General 12-1
12.2 ADV Overview. 12-1
12.3 Consolidation Reports. 12-2
12.4 ADV Phases 12-2
12.4.1 Input Phase. 12-2
12.4.2 Description Processing Phase 12-2
12.4.3 Expression Evaluation Phase 12-3
12.4.4 Checkword Evaluation Phase 12-3
12.4.4.1 Graphical Logic Verification 12-3
12.5 ADV Program 12-4
12.6 Input Processing 12-7
12.6.1 ADS PROM Code Data Report. 12-8
12.6.2 Symbol Table Data Report 12-9
12.6.3 Duplicate Names Report. 12-11
12.6.4 Duplicate Addresses Report 12-11
12.7 Description Processing 12-12

TABLE OF CONTENTS

Topic **Page**

12.7.1 Vital Input Report 12-12
12.7.2 Vital Output Report 12-13
12.7.3 NVSP Controls (NVSP to VSP) Report. 12-16
12.7.4 Vital Serial Input Message Parameters. 12-17
12.7.5 Vital Serial Output Message Parameters 12-19
12.7.6 Boolean Expression Report 12-20
12.7.7 Code System Indications Report 12-23
12.7.8 NVSP System Indications (VSP to NVSP) Report 12-24
12.8 Expression Processing 12-25
12.8.1 Expression Evaluation Report. 12-25
12.9 Checkword Processing 12-27
12.9.1 Memory constraints Report. 12-27
12.9.2 Timer Values Report. 12-29
12.9.3 Vital Serial Memory Constraints 12-30
12.9.4 VRD Checkword Report 12-31
12.9.5 RAM Address Report 12-35
12.9.6 Displacements and Increments Report. 12-37
12.9.7 Vital Serial Displacements and Increments 12-38
12.9.8 Vital Serial Offsets and Offset Increments 12-40
12.9.9 Shadow Bank Memory Offset Data Report. 12-42
12.9.10 Vital Serial Report Summary. 12-43
12.9.10.1 PD Sum Values 12-43
12.9.10.2 VSC Buffer and Memory Checkwords 12-46
12.9.10.3 Link Checkwords 12-47
12.9.10.4 Drop Address Values 12-49
12.9.10.5 Vital Serial Link and Block Assignments Report 12-49
12.9.11 DigiSAFE Report Summary 12-50
12.9.12 DigiSAFE Equipment ID Assignments 12-52
12.10 ADV System Messages / Errors 12-53
12.10.1 Abort and Error Messages 12-54

SECTION 13 – ADV Consolidation Reports 13-1

SECTION 14 – ADV Compare Program 14-1
14.1 Program Operation 14-1
14.2 Reports 14-2

TABLE OF CONTENTS

Topic **Page**

APPENDIX A – Application Data Verification (ADV) iVPI Data Sheet A-1
A.1 Introduction A-1
A.2 Safety Precautions A-1
A.3 iVPI ADV Consolidation Check Form A-4
A.4 ADV Troubleshooting Guidelines A-5
A.5 ADV Preparation A-6
A.6 ADV Consolidation Check. A-8
A.7 Conclusion A-54

APPENDIX B – ADV Compare Checklist B-1
B.1 Safety Precautions B-1
B.2 ADV Compare Preparation B-3
B.3 ADV Compare Check Form B-4
B.4 ADV Troubleshooting Guidelines B-5
B.5 Report Section B-6
B.5.1 SYMBOL TABLE REPORT, .SYM B-6
B.5.2 VITAL INPUT REPORT, .SYM B-6
B.5.3 VITAL OUTPUT REPORT, .SYM B-6
B.5.4 VITAL SERIAL INPUT MESSAGE REPORT, .SYM. B-7
B.5.5 VITAL SERIAL OUTPUT MESSAGE REPORT, .SYM. B-7
B.5.6 VITAL SERIAL CODEWORD REPORT, .SYM. B-7
B.5.7 VITAL SERIAL LINKS AND BLOCKS COMPARISON REPORT, .SYM B-7
B.5.8 DIGISAFE EQUIPMENT ID COMPARISON REPORT, .SYM B-8
B.5.9 BOOLEAN EQUATION REPORT, .SYM B-8
B.6 Conclusion B-8

LIST OF TABLES

Table No.	Title	Page
Table 1–1.	Warning Safety Precaution Headings and Location	1-1
Table 2–1.	Common Abbreviations and Glossary	2-3
Table 2–2.	Related Publications List	2-5
Table 4–1.	Valid Input Data Characters	4-1
Table 5–1.	DI Signature Assignments by Slot	5-7
Table 5–2.	Vital Output PROM Assignments by Slot	5-10
Table 5–3.	Vital Output Board Address Assignments by Slot	5-12
Table 5–4.	CRG Board ID Assignments by Slot	5-36
Table 5–5.	Non-vital Output States	5-39
Table 6–1.	Recommended INCLUDE File Organization for iVPI System Compiler	6-3
Table 6–2.	Recommended INCLUDE File Organization for iVPI System Comm Compiler	6-4
Table 6–3.	Recommended INCLUDE File Organization For NVSP Compiler	6-5
Table 6–4.	Shareable Network Files	6-6
Table 9–1.	iVPI Compiler Run Control Commands	9-3
Table 11–1.	NVSP Compiler Run Control Commands	11-3
Table 12–1.	Symbol Table Buffer Names	12-10
Table 12–2.	VRD Checkword Report Values	12-31
Table 12–3.	VRD Checkword Report Buffer Names	12-33
Table 13–1.	VPI CPU/PD Section Values	13-9
Table 13–2.	VSC Section Values for Point-To-Point VSC	13-9
Table 13–3.	VSC Section Values for Multidrop VSC	13-10
Table 13–4.	VSC Section Values for Code Rate Generator Boards	13-10

SECTION 1 – SAFETY PRECAUTIONS

1.1 SAFETY PRECAUTION MATRIX

Warning safety precautions are presented in Table 1–1.

Table 1–1. Warning Safety Precaution Headings and Location

Safety Precaution Heading	Found on page:
Overview Manual Must Be Read In Entirety	1–3
Modification of CAAPE and CAA Prohibited	1–3, 2–1
Use Only Alstom Vital Relay with VSP Board	1–3, 5–2
Protect Vital Output Equations With VRDFRNT-DI	1–4, 5–3
ACO Current Check Parameter Application	1–4, 5–31
Vital Communications Require Unique Link and Block Settings	1–5, 5–43, 5–54, 5–72, 9–24
CRG Status Parameter Application	1–5, 5–44, 9–32
DigiSAFE Communications Require Unique VSP Board ID	1–6, 5–60
DigiSAFE Communications Require Unique Zone Controller ID	1–6, 5–61
DigiSAFE Communications Require Unique Network IDs	1–6, 5–61
Protect Vital Timer Equations with VRDFRNT-DI	1–7, 5–98
Software Revision Control Must Be Maintained	1–7, 9–12
Unique Site ID Control Must Be Maintained	1–8, 9–12
Accurate Software Revision ID Control Must Be Maintained	1–9, 9–13
Unique System ID Control Must Be Maintained	1–10, 9–15
Link-Num Assignment Critical	1–10, 9–25
Block-Num Assignment Critical	1–11, 9–25
Programming VSP Board Overwrites FSSVT Settings	1–11, 13–2
FSSVT Modifications Must Be Verified	1–12, 13–2
FSSVT Modifications Must Be Field Tested	1–12, 13–3
FSSVT Passwords Must Be Protected	1–13, 13–3

Table 1–1. Warning Safety Precaution Headings and Location *(Cont.)*

Safety Precaution Heading	Found on page:
FSSVT Signature Values Must Be Verified	1–13, 13–3
Intended Safe Functionality of the iVPI System Must Be Verified	1–13, 13–3, A–1, B–1
iVPI Application Must Be Validation Tested	1–14, 13–4, A–2, B–1
iVPI Application Must Be Field Tested	1–14, 13–4, A–2, B–2
ADV Input Data Must be Verified Separately—Prior to ADV Process	1–14, 13–4, A–3, B–2
Verifier Must Be Different Than Designer	1–15, 13–5, A–2, B–2

1.2 SAFETY PRECAUTIONS

⚠WARNING

OVERVIEW MANUAL MUST BE READ IN ENTIRETY

The iVPI Overview manual (P2521A) must be read in its entirety prior to any operational and/or maintenance actions as it contains important safety messages and pertinent iVPI information.

Failure to comply may result in an unsafe condition or accident causing death or serious injury.

MODIFICATION OF CAAPE AND CAA PROHIBITED

No modification of CAAPE, CAA, or any of its component programs is allowed because any program change could compromise the safety performance of the system.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

USE ONLY ALSTOM VITAL RELAY WITH VSP BOARD

Only Alstom VRD relay (P/N 56001-787-05) is to be used with the Alstom iVPI system VSP board. Alstom products are designed to function within all-Alstom systems. The introduction of non-Alstom products into an Alstom iVPI system could have unintended and unforeseeable safety consequences.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

PROTECT VITAL OUTPUT EQUATIONS WITH VRDFRNT-DI

Relying on the status of the VRDFRNT-DI Vital input to, in effect, control Vital output devices without including the VRDFRNT-DI Vital input in the respective output equations does not provide fail-safe operation. The VRDFRNT-DI Vital input must be used as a constituent to the Vital output Boolean equations.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment. Customer application of VRDFRNT-DI in a non-vital manner is done so at the risk managed by the customer (Alstom Signaling takes no responsibility for that risk).

ACO CURRENT CHECK PARAMETER APPLICATION

The ACO current check parameter is calculated in a non-vital manner and shall not be applied as a fail-safe parameter.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

VITAL COMMUNICATIONS REQUIRE UNIQUE LINK AND BLOCK SETTINGS

Failure to properly assign, maintain and control unique Link and Block settings for Vital communications within iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

The message link and block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link and Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

CRG STATUS PARAMETER APPLICATION

The CRG status parameters are calculated in a non-vital manner and must not be applied as fail-safe parameters.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

WARNING

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE NETWORK IDS

The DigiSAFE iVPI IDs and the zone controller IDs must be assigned such that the values are unique to each other and to any other ID throughout the entire DigiSAFE network.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE VSP BOARD ID

A unique ID must be assigned to each iVPI VSP board in order to give each iVPI a unique identification in the network of DigiSAFE communications.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE ZONE CONTROLLER ID

A unique ID must be assigned to each zone controller in order to give each zone controller a unique identification in the network of DigiSAFE communications.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE NETWORK IDS

The DigiSAFE iVPI IDs and the zone controller IDs must be assigned such that the values are unique to each other and to any other ID throughout the entire DigiSAFE network.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

PROTECT VITAL TIMER EQUATIONS WITH VRDFRNT-DI

Vital Boolean and timer equations are evaluated in every one-second application cycle regardless of the state of the VRD. Therefore every timer equation must include the VRDFRNT-DI Vital input as a constituent in order to prevent the timer from running short and completing an evaluation of the equations prematurely.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

SOFTWARE REVISION CONTROL MUST BE MAINTAINED

Failure to properly version control iVPI system software and application data can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that strict revision control of the iVPI application data and system software be maintained so that the expected configuration in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

⚠WARNING

UNIQUE SITE ID CONTROL MUST BE MAINTAINED

Failure to properly assign, maintain and control unique Site IDs for iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that strict control of the Site IDs be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

⚠WARNING

ACCURATE SOFTWARE REVISION ID CONTROL MUST BE MAINTAINED

Failure to update and maintain the Software Revision IDs for **every** software change made to the application data and/or system software (even a re-compile done with no software changes) jeopardizes proper software revision control and can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that Software Revision IDs be changed with **every** software change, even a re-compile of unchanged software. Software Revision IDs shall be maintained so that software and application revision control is maintained and the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

⚠WARNING

UNIQUE SYSTEM ID CONTROL MUST BE MAINTAINED

Failure to properly assign, maintain and control a unique System ID for each iVPI system within the entire train control system can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that strict control of the System IDs be maintained so that the expected configuration of all iVPIs within the entire train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system, which deviate from Alstom's originally, delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

LINK-NUM ASSIGNMENT CRITICAL

Correct assignment of this *link-num* number is critical for system safety. The message link values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

Failure to properly assign, maintain and control unique Link settings for iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

WARNING

BLOCK-NUM ASSIGNMENT CRITICAL

Correct assignment of this *block-num* number is critical for system safety. The message block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

Failure to properly assign, maintain and control unique Block settings for iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

PROGRAMMING VSP BOARD OVERWRITES FSSVT SETTINGS

Programming an application into a VSP board erases and overwrites the previous application including all FSSVT settings. Any previous field updates to FSSVT settings will be overwritten and the FSSVT settings will be configured per the programmed application.

Failure to monitor and oversee these FSSVT values are as desired can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

FSSVT MODIFICATIONS MUST BE VERIFIED

All FSSVT modifications are safety-critical and must be verified, using the AlsDload program or the Application Data Verifier program within CAAPE, to determine whether the iVPI application PROM code data has been encoded as specified by the AlsDload FSSVT compiler.

Refer to Alstom Publication P2521A iVPI Product Overview Manual sections:

Application Verification:

The basis of the application of iVPI is to use a tool to configure the system hardware and software, as well as create the signaling logic for the vital application. The independent Application Data Verifier Tool, as well as associated procedures, must be run and performed prior to any iVPI application program be tested in field commissioning tests.

Proof of Logic (Primordial Logic Review):

The application of iVPI depends on experienced signaling engineers defining configurations and logic to be implemented for the interlocking application. While iVPI guarantees that logic and outputs, etc. are managed vitally, there is no intrinsic check on the correctness or completeness of the signaling logic as it is intended to meet the requirements of the transit or railroad application. It is a primary safety requirement that the logic produced for iVPI execution be independently verified as correct and complete through a "circuit check" type process. The check process must be performed by engineers knowledgeable in the requirements of the signaling rules that govern transit/railroad operation and independent from the engineering staff that produced the logic.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

FSSVT MODIFICATIONS MUST BE FIELD TESTED

All changes made to the FSSVT must be field tested to validate the intended timer values of any modified timers are observed to be correct in actual operation prior to the return of revenue service.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

FSSVT PASSWORDS MUST BE PROTECTED

FSSVT passwords shall be provided only to responsible personnel that have been properly trained in the FSSVT modification, verification, and validation process.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

FSSVT SIGNATURE VALUES MUST BE VERIFIED

Verify through Vital signatures that FSSVT values that were not intentionally changed have retained their original signature values.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

INTENDED SAFE FUNCTIONALITY OF THE IVPI SYSTEM MUST BE VERIFIED

The safety of the application logic as written is the responsibility of an experienced signaling engineer—CAAPE does not make any determination regarding the inherent safety of the logic equations that were entered.

Verifying the accuracy with which CAAPE converted the experienced signaling engineer's application data into PROM data structures is aided by CAAPE, but the signaling engineer must make a final determination using information supplied by CAAPE. CAAPE’s compilers are not themselves Vital programs. An additional independent process is needed to verify that the compile was done correctly. This process is required for all Vital applications.

An experienced signal engineer must verify the safety of the iVPI data and its application. It is the signaling engineer's responsibility to verify the correctness of the iVPI input data in that it accurately represents the intended safe functionality of the iVPI system. Furthermore, "verify the correctness" means that the signaling engineer (1) is required to compare the input and output data files to verify the CAA has operated correctly and (2) must test the iVPI application in its intended environment before it can be placed in revenue service.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

IVPI APPLICATION MUST BE VALIDATION TESTED

Prior to revenue service, validation testing must confirm all iVPI application logic is correct and consistent with application requirements.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

IVPI APPLICATION MUST BE FIELD TESTED

Field testing of a iVPI application is required before placing the location into revenue service. The customer's testing plan and safety plan define the testing requirements for the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

ADV INPUT DATA MUST BE VERIFIED SEPARATELY—PRIOR TO ADV PROCESS

Vital system operation requires that the Boolean equations in the Vital application logic must be written correctly, so that by executing the logic, the iVPI system operates safely in accordance with the rules of the transit or railroad authority.

The Application Data Verifier (ADV) output report provides a means to compare and verify equivalence between the input and the output application data.

However, the Application Data Verifier neither determines the safety suitability of the Boolean expression list nor determines the validity of certain encoded iVPI application data. The input data to the ADV process must be verified for safety separately, prior to the ADV process, and the safety and suitability of the input data is the responsibility of the experienced signaling engineer.

The ADV does, however, issue warnings and error messages as a result of non-vital data checking to alert the experienced signaling engineer to possible discrepancies.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

VERIFIER MUST BE DIFFERENT THAN DESIGNER

The experienced signaling engineer responsible for verification (the Checker or Verifier) using the ADV checklist and creating the report shall be independent from the signaling engineer responsible for designing (the Designer) the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

THIS PAGE INTENTIONALLY LEFT BLANK.

SECTION 2 – GENERAL

This is a reference manual for the iVPI Computer Aided Application (CAA) portion of Computer-Aided Application Programming Environment software (CAAPE).

2.1 SAFETY

⚠WARNING

MODIFICATION OF CAAPE AND CAA PROHIBITED

No modification of CAAPE, CAA, or any of its component programs is allowed because any program change could compromise the safety performance of the system.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

SAFETY INSTRUCTIONS

The responsibility for the underlying safety of Vital logic equations and other Vital application data belongs to the user (experienced signaling engineer). The user is responsible for verifying that the interlocking control equations developed by the program correctly specify the intended operation in a fail-safe manner.

2.2 INTENDED AUDIENCE

This manual is written for signaling application engineers and others who want detailed information on the iVPI CAA.

2.3 DOCUMENT CONVENTIONS

The following conventions are used in this manual:

bold	Bold text used in command lines represents information that should be entered exactly as shown (keywords).
bold italic	Bold italic text indicates icons to activate or selections in the menu tree.
italic	Italic text indicates a file name or a term.
[ ]	In command lines, square brackets indicate an option. To enter the option, type only the information inside the brackets. Do not type the brackets themselves.
|	When describing a menu selection, this character is used to separate consecutive menu item choices. For example, ***File*** **|** ***Exit*** means to open the File drop-down menu and select the Exit item.

2.4 COMMON ABBREVIATIONS AND GLOSSARY

Terms used throughout this manual are provided in Table 2–1.

Table 2–1. Common Abbreviations and Glossary

Term	Definition or Explanation
ACO	Vital AC Output board
.ADO	CAAPE output file
.ACR	CAAPE output file
ADV	Application Data Verifier
AF	Audio Frequency
BEX	Bus Expansion board
CAA	Computer-Aided Application
CAAPE	Computer-Aided Application Programming Environment
CRG	Code Rate Generator board
DBO	Double Break Output board
DI	Direct Input board
DPRAM	Dual-Ported Random Access Memory
Drawing Number	Alstom Part Number
EPROM	A programmable read-only memory device that is erasable using high intensity ultra-violet light
GTP	Genrakode Track Processor board
I/O	Input/Output
iVPI	Alstom’s Integrated Vital Processor Interlocking product
LDO	Lamp Drive Output board
LPC	Link Protocol Command
.LSV	CAAPE output file
.LVC	CAAPE output file
MMS	Alstom’s Maintenance Management System product

Table 2–1. Common Abbreviations and Glossary *(Cont.)*

Term	Definition or Explanation
NISAL	Numerically Integrated Safety Assurance Logic – the numeric algorithms that ensure the safety of iVPI's Vital software
NVI	Non-Vital Input boards
NVO	Non-Vital Output boards
NVSP	Non-Vital System Processor – the non-vital processor board in an iVPI system
PROM	Programmable Read-Only Memory – programmable memory devices that store firmware
RAM	Random Access Memory – this part of memory temporarily stores information that is constantly being changed in the computer; here, words may be stored (written) or read (retrieved) in any order at random
SBO	Single Break Output board
SSR	Solid State Relay
User	Experienced signaling engineer
.VCO	CAAPE output file
.VCR	CAAPE output file
VSC	Vital Serial Communications, Vital Serial Controller, includes CPU/PD, VSC, VRD, and IOB functions
VSOE	Vital Serial Over Ethernet, a method for vitally exchanging the states of Vital interlocking functions over a network
VSP	Vital System Processor – the Vital processor board in an iVPI system

2.5 RELATED PUBLICATIONS

Table 2–2. Related Publications List

Document No.	Title
P2326B	CenTraCode II-s Communications System Operation and Maintenance
P2512A	Computer Aided Application Programming Environment (CAAPE) User Manual
P2512B	AlsDload User Manual
P2512C	CenTraCode II-s CAA Reference Manual
P2512E	DataLogger User Manual
P2521A	iVPI® Integrated Vital Processor Interlocking Control System Product Overview Manual
P2521B	iVPI® Integrated Vital Processor Interlocking Control System Operation and Maintenance Manuals (Volumes 1 – 5)
P2346 series	Manuals for specific serial protocols

THIS PAGE INTENTIONALLY LEFT BLANK.

SECTION 3 – INTRODUCTION

iVPI CAA includes Vital and non-vital compilers and other low-level tools used to generate the EPROM files as well as other output and report files for iVPI systems.

Multiple versions of CAA can be made available through the top-level CAAPE package. Each version of iVPI CAA is identified by a part number 31746-*nnn*-00 Rev.*r*:

- *nnn* is the major version number (also referred to as the dash number).
- *r* is the minor version revision letter.

In general, the major version number changes with any change to the Vital system software which is distributed as part of the CAA package; the revision letter changes when new releases are made to fix bugs or add features that do not affect the Vital system software or the EPROM data structures it uses.

A particular iVPI CAA can be identified by its version information, for example 600A. Part numbers, also referred to as drawing numbers, are discussed in greater detail under SECTION 4.5 – Drawing Numbers.

3.1 THEORY OF OPERATION

Each iVPI CAA includes multiple compilers:

- The iVPI Compiler, also referred to as the Vital compiler, processes the Vital application and creates the EPROM data for the main subsystem of the VSP board.
- The iVPI Comm Compiler, also referred to as the Vital Comm compiler, processes the network communications application and creates the EPROM data for the Comm subsystem of the VSP board.
- The NVSP Compiler, also referred to as the non-vital compiler, processes the non-vital application and creates the EPROM data for the NVSP boards.

All compilers read user-created input data describing the application. The top-level CAAPE package allows the input data to be created either indirectly using a graphical interface or directly through manual editing of text files, but in either case the compiler ultimately reads text-based input records. This manual describes the format of the text records read by the compilers.

An Application Data Verifier (ADV) Program is used to verify that the Vital compiler has produced Vital application data structures consistent with what was originally entered by the user, for example that the compiler did not change the contents of a logic equation so that it no longer operates as the user intended. An ADV Compare (ADVC) Program can be used to compare the outputs of two ADV sessions to identify changes made to the Vital application. This manual includes descriptions of ADV and ADVC operation and reports. Also included is an ADV Section that guides the user through a thorough verification of the compiler output. An ADV data sheet is provided for direction and proof of the verification process. This comprehensive checklist, when successfully completed with no discrepancies, serves as proof the iVPI compiler program correctly encoded the user's application data. See APPENDIX A – Application Data Verification (ADV) iVPI Data Sheet.

SECTION 4 – GENERAL RULES AND NOMENCLATURE

4.1 RECORDS

All inputs to the compiler program are referred to as records. A record consists of readable text on one or more lines of the file; each line is terminated by a carriage return and line feed combination. Unless otherwise indicated, most records must fit on a single line. When a specific category of record can go beyond a single line, the user must follow any record-specific rules for using multiple lines (for example, line continuation characters).

4.2 DEFAULT VALUES

In many cases, if the user does not provide all the input information for a given situation, the CAA programs provide default values for the missing data. A default value is defined as the action a CAA program assumes if the user does not fully specify all the options for a given situation.

4.3 FREE FORM DATA

All input to the programs in the CAA package is free form, except where noted. Free form data means that the user can enter data anywhere in the columns of the data record. Comment records are an exception to this, they have additional criteria.

Valid input characters for data records are listed in Table 4–1. Delimiters are characters which separate the data fields in a record. Specific delimiter characters to be used depend on the record type and are included in the record format descriptions in later sections.

Table 4–1. Valid Input Data Characters

Character Group	Characters
Alphabetic	A through Z (upper case only)
Numeric (decimal)	0 through 9
Alphanumeric	A through Z (upper case only) and 0 through 9
Hexadecimal	0 through 9 and A through F (upper case only). Numbers in hexadecimal base are represented by the prefix '0x'.
Delimiters	Depend on record type

4.4 COMMENT RECORDS

The major exception to free form input data is the comment data record. Any data record with an asterisk in the leftmost column (column 1) is considered a comment record. A comment record enables the user to place comments anywhere within the compiler input file, and these can be printed on the output listings when running the appropriate programs.

The iVPI (Vital) Compiler produces a warning message if it encounters a comment embedded within a Boolean equation, since accidental commenting of equation records changes the operation of the equation.

4.5 DRAWING NUMBERS

A number of the data records have a field for drawing numbers (part numbers). The proper format for a drawing number is as follows:

- Key Number - five digit number, or SK followed by a four digit number.
- Dash Number - one to three digit number (Alstom always uses three digits).
- Group Number - a dash followed by a two digit group number or "GR" followed by a one or two digit number (Alstom always uses two digits).

NOTICE

The use of “GR” to indicate the group number is from an older drawing number format. The Alstom format uses a dash followed by a two digit group number.

Be aware that the iVPI CAA uses the older format for compatibility with existing files.

Some records also include revision information in the form of “, REV ” or "Rev. " followed by the revision data.

Examples of Alstom drawing numbers:

31751-014-00 Rev. L

39780-003-01

59473-749-04, REV B

Examples of older drawing numbers using "GR":

31681-001 GR. 01

31682-1 GR 1

33035-123GR01, REV A

SK8193-1 GR 00

NOTICE

Drawing numbers shown in this manual are for illustration only, and are not necessarily actual Alstom numbers.

4.6 PARAMETER NAMES

All parameter names to be used in expressions are predefined prior to the Boolean expression list. Therefore, each parameter name used in an expression can be checked and verified. The names of actual I/O and any names assigned to internal parameters must be unique.

Parameter names can be up to 16 characters in length; any character can be used with these exceptions:

- Must not contain any of the delimiter characters for any records where they may appear, including:
  - Space [ ]
  - Comma [,]
  - Left parenthesis [( ]
  - Right parenthesis [ )]
  - Equal sign [=]
  - Ampersand [&]
  - Plus sign [+]
    - however, wire names may contain the plus sign [+]
- Must not begin with the three-character sequence:
  - period capital N period [.N.]
- Must not have the 'at' symbol [@] as the first character
- Must not be reserved by the system (see Section 4.6.1)

4.6.1 System Reserved Names

Two system-reserved names have special meanings and must not be assigned as parameter names.

- PERMONE – indicates values that are always True
- PERMZERO – indicates values that are always False

Depending on the compiler, certain variable names have predefined CAA usages or are generated by the compiler and are therefore reserved names.

Compilers which use VSOE2 generate LinkOk parameters for each VSOE link. The names of these parameters are:

- VSOENNN-LINKOK – where NNN will be the link number (range 1 to 200) of the link they represent.

These names are reserved and cannot be used.

4.7 BOARD IDENTIFIER NAMES

Certain types of boards must be identified by a user-assigned name in specific sections of input data.

Some boards such as NVSP must be given a numeric and a text identifier; both are defined in a record in the main Vital and non-vital application files.

For example, the record format for NVSP is:

NVSP num ID= name

- *num* is a number from 1 to the number of NVSP boards in the system.
- *name* is the text name assigned to the board.

For example:

NVSP 1 ID = CONTROL PANEL NVSP

The numeric identifier must correspond to the board number assigned in the non-vital application's settings in CAAPE. The same number must appear in the hardware slot assignment record for the board:

For example:

SLOT 4 = NVSP BOARD 1, 59473-938GR00

When the board appears in certain input sections, it is identified by its name.

For example, in the VSP Communications section,

SOURCE = CONTROL PANEL NVSP

indicates that the NVSP-to-VSP message is being sent by this board.

SECTION 5 – IVPI APPLICATION RULES

This section describes the aspects of iVPI rules and operation that have a bearing on iVPI CAA input, including:

- Hardware Application Rules
- VSP-NVSP Communications
- GTP Communications
- CRG Communications
- Networking
- Network Communications (VSP)
- Network Communications (NVSP)
- Vital Serial Links and Blocks
- Non-Vital Serial Communications
- Data Logging
- Vital Logic
- Non-Vital Logic

5.1 HARDWARE APPLICATION RULES

Hardware application rules include the rules of application of the boards in iVPI modules as enforced by the Compiler Programs. Each board has a unique set of application rules that the Compiler Programs enforce.

⚠WARNING

USE ONLY ALSTOM VITAL RELAY WITH VSP BOARD

Only Alstom VRD relay (P/N 56001-787-05) is to be used with the Alstom iVPI system VSP board. Alstom products are designed to function within all-Alstom systems. The introduction of non-Alstom products into an Alstom iVPI system could have unintended and unforeseeable safety consequences.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

5.1.1 Output Protection

WARNING

PROTECT VITAL OUTPUT EQUATIONS WITH VRDFRNT-DI

Relying on the status of the VRDFRNT-DI Vital input to, in effect, control Vital output devices without including the VRDFRNT-DI Vital input in the respective output equations does not provide fail-safe operation. The VRDFRNT-DI Vital input must be used as a constituent to the Vital output Boolean equations.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment. Customer application of VRDFRNT-DI in a non-vital manner is done so at the risk managed by the customer (Alstom Signaling takes no responsibility for that risk).

The primordial logic should be designed to assure that failures in internal and external circuitry, including the VRD Relay and VRD Repeater Relays, result in known safe conditions. All iVPI output control equations should be evaluated by an experienced signal engineer to include a VRDFRNT-DI parameter to ensure that all outputs—for example signals and Vital serial parameters—are placed in a restrictive state in the event of a system failure including a failure in the VRD Relay or VRD Repeater Relay circuitry external from the iVPI system.

Every Vital system requires at least one B relay which is operated by the VRD and through whose front contacts all the energy for the Vital outputs is broken. This relay must be, and must only be replaced by, an Alstom VRD Relay, part number 56001-787-05, 100 ohm B relay.

A front contact from the VRD Relay must be fed back into the iVPI system as a Vital input for use in the application. For example, to prevent Vital timers from starting when the VRD is de-energized. The name of this Vital input may be VRDFRNT-DI.

NOTICE

The front contact used as the Vital input is also available to supply energy to Vital outputs.

5.1.2 Board Placement

The hardware of an iVPI system is comprised of a main (System) module and up to three Expansion modules, numbered 1 through 3.

An iVPI module has up to 21 slots. Within a System module, the iVPI motherboard may be split to allow multiple independent iVPI systems to exist within a single chassis. The board placement rules for a given system are relative to the starting slot of its section of the module. For example, saying that a VSP board is placed in slot 1 and takes up two slots in a system that starts in slot 4 of the module means that this VSP board is in slots 4 and 5 of the module. Similarly, the rule that there can be no more than one VSP in a system applies only to the section of the module occupied by that system. If there is more than one system in a module with a split motherboard, each system can have its own VSP board.

NOTICE

Unless otherwise specified, all slot numbers listed below are relative to the starting slot of the system within a module with a split motherboard.
Any rules relative to maximum numbers of boards refer to the module section occupied by the system. “Module” refers to the entire module if there is no split motherboard or to a system’s module section if there is.

5.1.2.1 Board Placement Rules

An iVPI module has up to 21 slots; a system in the module can have from 1 to 21 slots depending on whether the motherboard is split. The following general rules must be followed when placing PC boards in slots.

- If there is a Vital application, a VSP board must be placed in slot 1 of the System module and a BEX board must be placed in slot 1 of each Expansion module.
- If there are non-vital applications, each application must be associated with an NVSP board placed between slots 2 and 8 of the module.
- Vital input boards in slot 14 of the System module and Expansion modules 1 and 2 are not available if Expansion module 2 or 3 is used. Vital input boards in slot 15 of the System module and Expansion module 1 are not available if Expansion module 3 is used.
- GTP and CRG boards cannot be placed in slots 10-13 or 20-21 of an Expansion module.
- iVPI is designed to minimize the need to manually wire board address jumpers. Boards are automatically assigned hardware signatures and addresses based on the slots in which they have been placed. Since board signatures and addresses are automatically assigned by slot position, the user must place boards in such a way that certain hardware rules such as uniqueness of vital I/O board signatures are not violated. These hardware rules are described in subsequent sections.

NOTICE

An NVSP board with a P3 Interface (P/N 31166-475-01) requires two slots.

Non-Vital Input boards must be grouped together and not interleaved with Non-Vital Output boards.
Likewise, Non-Vital Output boards must be grouped together and not interleaved with Non-Vital Input boards.

5.1.2.2 Board Address Assignment

Board addresses are always assigned by the Compiler Program, and cannot be overridden.

The Board Report for each module supplies the wiring assignment for each board. See SECTION 9.1.2.1 – COMPILER RUN CONTROLS Records for information on how to generate board reports.

5.1.3 Network Boards

VSP and NVSP boards contain two network devices, designated ENET1 and ENET2, and separate processors to control network communications. On each board, the Main processor controls logic, I/O and message protocol operations and the Comm processor controls the actual network communications.

VSP boards have separate Main and Comm applications produced by separate compilers. Main and Comm applications can be programmed directly into EPROM or downloaded using the AlsDload tool through serial or USB connections.

NVSP boards have a single Main application; the Comm software is generic and gets its commands from the Main processor. The Main application and the Comm software can be downloaded using the AlsDload tool.

See the sections on Network Communications for details.

5.1.4 Vital Input Boards

There are a maximum of 14 Vital direct input (DI) boards allowed per module.

All direct input boards require a signature. This signature is always assigned by the Compiler Program based on board slot position, and cannot be overridden. The signatures available for DI boards are designated by the letters C, D, E,..., O, P. Each board in a module must have a unique signature; since signatures are assigned by slot, boards must be placed so that there are no duplicate signatures in a module.

Refer to Table 5–1 when placing DI boards to ensure that no signature is duplicated within a module.

Table 5–1. DI Signature Assignments by Slot

Slot	System Module Signature	Expansion Module 1 Signature	Expansion Module 2 Signature	Expansion Module 3 Signature
2	P	P	P	P
3	O	O	O	O
4	N	N	N	N
5	M	M	M	M
6	L	L	L	L
7	K	K	K	K
8	J	J	J	J
9	I	I	I	I
10	H	H	H	H
11	G	G	G	G
12	F	F	F	F
13	E	E	E	E
14	D	D	D	P
15	C	C	P	O
16	P	P	O	N
17	O	O	N	M
18	N	N	M	L

Table 5–1. DI Signature Assignments by Slot *(Cont.)*

Slot	System Module Signature	Expansion Module 1 Signature	Expansion Module 2 Signature	Expansion Module 3 Signature
19	M	M	L	K
20	L	L	K	J
21	K	K	J	I

A Vital direct input board contains 16 input ports; each port requires a decimal integer port identification, the number of cycles of forgiveness for the port, and the wiring assignments for the two wires that the port is attached to through the back plug coupler.

Port numbers are assigned to the board from top to bottom, numbered 1 to 16.

To give a measure of protection from 'noisy' inputs, a method of specifying a degree of forgiveness is devised. If an input is read that the iVPI determines is neither TRUE nor FALSE, the 'cycles of forgiveness' data field determines whether the iVPI assigns it a FALSE value immediately, or assigns the input the value it had from the previous cycle.

NOTICE

It is advisable to give all inputs one cycle of delay, minimum. If two cycles can be tolerated, this is even better. This function acts only as a noise filter, so that an input that is truly OFF registers as a FALSE value immediately. The value from the previous cycle is only used if an input is read as neither ON nor OFF.

Although it is not a rule which is enforced by the Compiler, it is strongly advised that the VRD front contact input be assigned two cycles of delay, to prevent noise from resetting the system.

The method of implementation of cycles of forgiveness follows.

- If 0 cycles of forgiveness is assigned to a port, then an undefined value read for the input causes a value of FALSE to be immediately assigned to the input.
- If 1 cycle of forgiveness is assigned, then the first time an undefined value is read for an input it is assigned the value it had on the previous cycle.
  - If the input is read on the next (2^{nd}) cycle in an unknown state, the input is assumed FALSE until a valid TRUE state is read.

The same concept applies when two cycles of forgiveness are assigned, for which the iVPI 'forgives' two noisy inputs in a row, but assigns a FALSE value if the input remains garbled thereafter. The cycles of forgiveness is specified by the field '(nCD)' following the integer port identifier, where 'n' is the number of cycles of forgiveness, 0, 1 or 2. If this data field is not specified for an input port, 0 cycles of forgiveness is assumed.

5.1.5 Vital Output Boards

5.1.5.1 Output PROMs

Vital output boards of all types are assigned unique PROM data. This data is different for every output in a system; therefore, each output board PROM is assigned a unique identifier.

The PROM contains the output check data for the board. Each PROM used in a system is unique, and the data is always assigned by the Compiler Program. This data is tied to a specific set of outputs and their address to prevent failures from compromising the system safety. Use Table 5–2 when placing vital output boards to ensure that no PROM identifier is duplicated within a system.

Table 5–2. Vital Output PROM Assignments by Slot

Slot	System Module Signature	Expansion Module 1 Signature	Expansion Module 2 Signature	Expansion Module 3 Signature
2	1	21	1	21
3	2	22	2	22
4	3	23	3	23
5	4	24	4	24
6	5	25	5	25
7	6	26	6	26
8	7	27	7	27
9	8	28	8	28
10	9	39	19	39
11	10	30	10	30
12	11	31	11	31
13	12	32	12	32
14	13	33	13	33
15	14	34	14	34
16	15	35	15	35
17	16	36	16	36
18	17	37	17	37

Table 5–2. Vital Output PROM Assignments by Slot *(Cont.)*

Slot	System Module Signature	Expansion Module 1 Signature	Expansion Module 2 Signature	Expansion Module 3 Signature
19	18	38	18	38
20	19	29	9	29
21	20	40	20	40

Two vital output boards are paired when one board uses the low part and the other board uses the high part of the same output address, allowing two 8-port boards to use a single 16-bit address. Board pairs are assigned based on their slot positions: boards in adjacent slots are automatically paired. No more than one board in a module can be unpaired, but it is not necessary that the unpaired board be the last one in the module.

Assign vital output boards according to Table 5–3 so that no more than one board is unpaired within a module. It is not necessary that paired boards be of the same type.

Table 5–3. Vital Output Board Address Assignments by Slot

Slot	Address	High/Low
2 3	30002	L H
4 5	30004	L H
6 7	30006	L H
8 9	30008	L H
10 11	3000A	L H
12 13	3000C	L H
14 15	3000E	L H
16 17	30010	L H
18 19	30012	L H
20 21	30014	L H

5.1.5.2 Lamp Drive Output Board

Lamp drive output boards (LDO type) contain eight outputs, divided into two groups of four for power supply wiring. Each group of four requires a GROUP data record to specify the positive and negative (common) supplies for the four ports in each group.

LDO boards are logically grouped in pairs for addressing purposes (since 16 outputs can be addressed at a time). Boards can be paired via the slot number reference following the words 'LDO BOARD' on the slot assignment record. The paired boards must be in adjacent slots in the module, but paired boards for a single address do not have to be the same Vital output board type. There can be one unpaired Vital output board per module.

Each output port used requires an I/O port data record to specify the integer identification of the output port. The numbers are assigned from top to bottom on the board.

The data record must also include the wire name of the negative wire that the port is attached to through the plug coupler. Each output port requires the specification of the negative only, since the four output ports in a group are referenced to the positive supply specified on the GROUP data record for the four. Two plug coupler pins are supplied for each output with one side always referenced to the group positive energy and this can be used as a tie point, if desired.

If the filament is to be checked for a Lamp drive output, the option following the port number is '(COLD-LO-CK)' or '(HOT-LO-CK)' to indicate the cold and hot filament check results for that port. The COLD-LO-CK is the cold filament check, which is only true if the filament is intact and the output is off. The HOT-LO-CK is a hot filament check, and represents a parameter that is true if the filament is intact and the bulb is lit. Each filament check value used must be assigned a unique name; it cannot be the name of the output being checked.

The port numbers are assigned to the board from top to bottom, numbered 1 to 8.

NOTICE

If two boards are paired, the ports are still identified 1 to 8 for the first board and 1 to 8 for the second board.

Lamp drive outputs can be driven steady, flashed, or alternate flashed. Alternate Flashing (also referred to as alt. flash) on LDO boards is not available in all CAAs.

Alternate flashing allows the application to flash Vital outputs so that the flashing is out of phase with the normal method of flashing. In a 1-second cycle, a normal flashing Vital output turns ON for the first half-second and OFF for the second half-second; an alternate flashing Vital output turns OFF the first half-second and ON the second half-second.

If any Vital output in a system can flash, however, then all lamp drive outputs are susceptible to inadvertent flashing when they are supposed to be on steady.

The customary restriction in the United States is that a flashing signal is more permissive than a steady one. The iVPI Compiler includes an option to generate equations that protect an output by causing it to turn off if it inadvertently flashes. (If a system has no flashing outputs, no protection is necessary.)

The customary restriction in European applications is that a steady signal is more permissive than flashing. In this case a system needs to be protected from an output accidentally going steady due to a failure when it is supposed to be flashing. There are no provisions in the iVPI CAA to automatically protect against this condition; however, equations can be written to enforce the protected condition.

Protection against inadvertent flashing requires the use of special "state" parameter data for flashing and alternate flashing outputs. The Compiler Program uses the VITAL OUTPUT FLASHING option to determine whether to allow the state parameters to be calculated.

If a steady signal is more permissive than flashing in this application, and protection against inadvertent steady outputs is required, use the RESTRICTIVE FLASHING ASPECTS option. The CAA does NOT automatically generate the required equations, but makes available a flash state parameter for use in the application. Alternate flashing is not available for restrictive flashing.

5.1.5.2.1 Data Records for Vital Flashing

Data records that describe Vital flashing for each lamp drive output in a system with flashing may indicate:

1. Unused - never on steady, flashing, or alternate flashing, thus not protected.
2. Never flashes or alternate flashes, never protected - there are no conditions under which this output flashes or alternate flashes, but even if it accidentally occurs, it is not critical.
3. Never flashes or alternate flashes, but protected - this output should never flash or alternate flash, and the steady state must always be protected from flashing/alternate flashing.
4. Never flashes or alternate flashes, protected, and under certain conditions not protected - although the output should not flash or alternate flash, there are certain conditions, as specified by a single equation, which bypasses the protection logic.
5. Steady and flash and/or alternate flash, not protected - this is not common; an output has conditions for being on steady and for flashing/alternate flashing, but flashing and alternate flashing are not considered more permissive than steady.
6. Steady and flash and/or alternate flash, protected - an output has conditions for being on steady and for flashing/alternate flashing, and flashing and alternate flashing are more permissive conditions, so the output must be protected from inadvertent flashing and alternate flashing.
7. Steady and flash and/or alternate flash, protected, and under certain conditions not protected - although the output should flash or alternate flash under some conditions, when on steady it must be protected from inadvertent flashing and alternate flashing; there are certain conditions, as specified by a single equation, which bypass the protection logic.
8. Flashes and/or alternate flashes, not protected - this is not common; an output has conditions for flashing and/or alternate flashing, but is never on steady, so it does not require protection.

Some examples of each type of lamp drive output:

Example 1 - unused output

1, = wire-name

Example 2 - never flashes or alt. flashes, never protected:

2,ldo-name2 = wire-name

2 (NONPROT), PERMONE

* Example 2 - alternative form; this format is also accepted, but with a warning; the Compiler inserts the '(NONPROT), PERMONE' data:

2,ldo-name2 = wire-name

Example 3, never flashes or alt. flashes but automatically protected:

3,ldo-name3 = wire-name

3 (ON), ldo-on3

Example 4 - no flash or alt. flash, automatically protected, sometimes not protected:

4,ldo-name4 = wire-name

4 (ON), ldo-on4

4 (NONPROT), ldo-nonprot4

Example 5 - steady and flash and/or alt. flash, not automatically protected:

5,ldo-name5 = wire-name

5 (FLASH), ldo-flash5 and/or 5(ALT-FLASH), ldo-aflash5

5 (NONPROT), PERMONE

* Example 5 - alternative form; this format is also accepted, but with a warning; the compiler inserts the '(nonprot), PERMONE' data:

5,ldo-name5 = wire-name

5 (FLASH), ldo-flash5 and/or 5(ALT-FLASH), ldo-aflash5

Example 6 - steady and flash and/or alt. flash, automatically protected:

6,ldo-name6 = wire-name

6 (ON), ldo-on6

6 (FLASH), ldo-flash6 and/or 6(ALT-FLASH), ldo-aflash6

Example 7 - steady and flash and/or alt. flash, automatically protected, sometimes not protected:

7,ldo-name7 = wire-name

7 (ON), ldo-on7

7 (FLASH), ldo-flash7 and/or 7(ALT-FLASH), ldo-aflash7

7 (NONPROT), ldo-nonprot7

Example 8 - flash and/or alt. flash only, not automatically protected:

8, = wire-name

8 (FLASH), ldo-flash8 and/or 8(ALT-FLASH), ldo-aflash8

Examples 1-8 represent all of the allowable combinations of flashing, alt. flashing and protection for lamp drive output ports. The order in which the parameters are named must be maintained according to these examples. Each name must be unique, and cannot be used for more than one output port.

In the next example, Example 9, if a parameter is defined as a CURRENT RESULT type, this applies to the usage of the parameter in Boolean equations; the parameter does NOT need to be listed in the CURRENT RESULT SECTION; defining it for the lamp drive output port is sufficient.

The four parameters are as follows:

- **ldo-name** is the name of the actual lamp drive output port. If the output is protected, the Compiler Program generates the equation for which this is a result. If it is not protected, the user must generate the appropriate equation. This entity must be given a name for all conditions except: 1, an unused output; or 2, an output which flashes but has no steady state.
- **ldo-on** is the result of the equation which defines the conditions for turning the output on steady. If the equation for this parameter is true and the 'ldo-flash' equation is true, the output flashes. If the 'ldo-on' equation is true but the flash equation is false, the output is on steady. This parameter is treated as a CURRENT RESULT type, but it does not need to be listed in the CURRENT RESULT SECTION. This parameter name is required unless the output is not used, or not automatically protected. If used, however, the user must supply the equation for which it is a result.
- **ldo-flash** is the result of the equation which defines the conditions for flashing the output. The output only flashes if the equation for this parameter becomes true. This parameter is treated as a CURRENT RESULT type, but it does not need to be listed in the CURRENT RESULT SECTION. This parameter name is only needed if the output is required to flash. If it is used, however, the user must supply the equation for which it is a result.
- **ldo-aflash** is the result of the equation that defines the conditions for alternate flashing the output. The output only alt. flashes if the equation for this parameter becomes true. This parameter is treated as a CURRENT RESULT type, but it does not need to be listed in the CURRENT RESULT SECTION. This parameter name is only needed if the output is required to alternate flash. If it is used, however, the user must supply the equation for which it is a result.
- **ldo-nonprot** is the result of the equation which defines the conditions for the steady output to NOT be protected, for instance, when the aspect is changing. This parameter name is required unless the output is not protected at all, or is to be protected all the time. This parameter is treated as a CURRENT RESULT type, but it does not need to be listed in the CURRENT RESULT SECTION. The user must supply the equation for which this is a result.

For restrictive flashing aspects, the ldo-flash parameter only is made available.

LDO 'state' parameters are available in addition to the previous entries and the filament check values '(HOT-LO-CK)' and '(COLD-LO-CK).' The three 'state' parameters are only available for lamp drive outputs in a system which has Vital flashing. These values are present for every lamp drive output, and they represent the actual state of the lamp drive output from the previous 1-second cycle, either on, flash or something else. The 'state' parameters cannot be used in their complemented state, that is, '.N.name' in an equation. The 'state' parameters do not have a True False state, only a specific value and thus cannot be viewed in non-vital diagnostic utilities with a proper representation of a True/False state; the actual 32-bit number must be queried and compared with the expected 32-bit value as defined in the application LVC file in order to determine if the output is calculated to be in a specific 'state’.

Example 9, all parameters, including 'state' parameters:

1,ldo-name1 = -wire-name

1 (ON), ldo-on1

1 (FLASH), ldo-flash1

1 (ALT-FLASH), ldo-aflash1

1 (NONPROT), ldo-nonprot1

1 (HOT-LO-CK), ldo-hotck1

1 (COLD-LO-CK), ldo-coldck1

1 (ON-STATE), ldo-on-state1

1 (FLASH-STATE), ldo-flash-state1

1 (ALT-FLASH-STATE), ldo-aflash-state1

In this example the additional named parameters are as follows:

- **ldo-hotck** is the name of the hot filament check parameter.
- **ldo-coldck** is the name of the cold filament check parameter.
- **ldo-on-state** is the state of the output during the previous 1-second cycle. This parameter is the proper value if the output was actually on steady.
- **ldo-flash-state** is the state of the output during the previous 1-second cycle. This parameter is the proper value if the output was flashing for the last cycle.
- **ldo-aflash-state** is the state of the output during the previous 1-second cycle; this parameter is the proper value if the output was alternate flashing for the last cycle

The parameters ('on-state', 'flash-state' and ‘alt. flash-state’) are optional, and need only be defined if the user wishes to use the value in some equation. They are a unique kind of parameter; neither current result nor self-latched, and the Compiler Program assigns them accordingly. State parameters are available for all lamp drive outputs in a system which specifies the VITAL OUTPUT FLASHING option.

NOTICE

There is no parameter for the OFF state since maintaining an output in its off state is a Vital function of the iVPI system; therefore, an incorrect output for a false output parameter results in loss of power to all outputs. The false output parameter is a sufficient indication of an output's off state.

For restrictive flashing aspects, ldo-flash-state is true when the output is not on for the second half of a 1 second cycle. Ldo-on-state is not available. For restrictive flashing aspects, alternate flashing is not available.

The flash rate for Vital flashing is always 1/2 second on, 1/2 second off.

The flash rate for Vital alternate flashing is always 1/2 second off, 1/2 second on.

5.1.5.2.2 Equations for Vital Flash Protection

Equations are generated by the Compiler Program when a lamp drive output is protected against inadvertent flashing.

The parameters 4ECL-ON, 4ECL-FLASH, 4ECL-ALTFLASH and 4ECL-ASPCHG are examples of names assigned in the input file with the options (ON), (FLASH), (ALT-FLASH) and (NONPROT), and these are used as equation results prior to the generated equations. A name is also assigned to the actual output. The equation for this name is generated by the Compiler. All other parameter names shown are generated by the Compiler Program.

For example:

```
@LDO30008-08-PST =
        (@LDO30008-08-ST2 +
        @LDO30008-08-ON * @LDO30008-08-PST)
*
@LDO30008-08-ST2 =
        (@LDO30008-08-ST1 +
        4ECL-ASPCHG +
        4ECL-FLASH +
        4ECL-ALTFLASH)
*
@LDO30008-08-ST1 =
        (.N.4ECL-ON +
        4ECL-FLASH +
        4ECL-ALTFLASH)
*
4ECL-LDO =
        (4ECL-ON * @LDO30008-08-PST)
```

5.1.5.2.3 Data Records for LDO Board

Data records for LDO boards include the allowable options.

For example:

```
SLOT 16 = LDO BOARD (15), 31166-431 GR01
GROUP A                 = B12-L , N12-L
1 2EAG-LDO                          = 2EAGE
1(HOT-LO-CK) 2EAG-HCK
1(COLD-LO-CK) 2EAG-CCK
2 2EAY-LDO                          = 2EAYE
2(HOT-LO-CK) 2EAY-HCK
2(COLD-LO-CK) 2EAY-CCK
3,                                          = 2EBLE
4,                                          = 4EAGE
GROUP B                 = B12-L , N12-L
8 4ECL-LDO                          = 4ECLE
8(ON) 4ECL-ON
8(FLASH) 4ECL-FLASH
8(ALT-FLASH) 4ECL-ALTFLASH
8(NONPROT)      4ECL-ASPCHG
8(HOT-LO-CK) 4ECL-HCK
8(COLD-LO-CK) 4ECL-CCK
```

In this example, ports 1 and 2 are assigned with their hot and cold filament check parameters. Ports 3 and 4 are prewired spares, and ports 5 through 7 are unused. All group wiring is defined. The parameters for protected Vital output flashing are defined for port 8.

5.1.5.3 Single Break Output Board

Single break output boards (SBO type) contain eight outputs, divided into two groups of four for power supply wiring. Each group of four requires a GROUP data record to specify the positive and negative (common) supplies for the four ports in each group.

SBO boards are logically grouped in pairs for addressing purposes (16 outputs can be addressed at a time). Boards can be paired via the slot number reference following the words 'SBO BOARD' on the slot assignment record. The paired boards must be in adjacent slots in the module, but paired boards for a single address do not have to be the same Vital output board type. There can be one unpaired Vital output board per module.

Each output port used requires an I/O port data record to specify the integer identification of the output port. The numbers are assigned from top to bottom on the board.

The data record must also include the name of the positive wire that the port is attached to through the plug coupler. Each output port requires the specification of the positive only, since the four output ports in a group are referenced to the negative supply specified on the GROUP data record for the four. Two plug coupler pins are supplied for each output, however, but one side is always referenced to the group negative energy and thus can be used as a tie point, if necessary.

If the logical value of the absence-of-current-detecting (AOCD) check circuit is to be used, another I/O port data record is required to provide the symbolic name of the AOCD circuit. The second port data record includes the number of the port being referenced, and the option (CK) following the number. Each AOCD check value used must be assigned a unique name and it must differ from the name of the output being checked. This (CK) value is used to detect the presence or absence of load current on an output.

The port numbers are assigned to the board from top to bottom, numbered 1 to 8.

NOTICE

If two boards are paired, the ports are still identified 1 to 8 for the first board and 1 to 8 for the second board.

Single break outputs can be driven steady or flashed. The flash condition is on for 1/2 second and off for 1/2 second. If any Vital output in a system is required to flash, then all Vital outputs are susceptible to inadvertent flashing when they are supposed to be on steady. In many cases a flashing lamp drive output may be a more permissive aspect than a steady output, so there is a means to protect lamp drive outputs. See Lamp Drive Output Board Application Rules for details.

In order to flash Vital outputs, the VITAL OUTPUT FLASHING option must be enabled.

5.1.5.3.1 Data Records for Vital Flashing

There are fewer Vital output flash options for single break outputs than lamp drive outputs. In addition, there are no CAA-generated equations associated with single break outputs; Boolean equations must be written to detect incorrect conditions and downgrade accordingly.

Vital flashing options include:

1. Unused - output never on steady or flashing.
2. Steady only - there are no conditions under which this output flashes and even if it accidentally occurs, it is not critical (this is the normal case of a single break output).
3. Steady and flash - an output has conditions for being on steady and for flashing.
4. Flashes only - an output has conditions for flashing, but is never on steady.

Four examples of single break output are shown.

Example 1, unused output:

```
1, = wire-name
```

Example 2, steady only:

```
2,sbo-name2 = wire-name
```

Example 3, steady and flash:

```
3,sbo-name3 = wire-name
3 (FLASH), sbo-flash3
```

Example 4, flash only:

```
4, = wire-name
4 (FLASH), sbo-flash4
```

NOTICE

In the following descriptions, if a parameter is defined as a CURRENT RESULT type, this applies to the usage of the parameter in Boolean equations; the parameter does NOT need to be listed in the CURRENT RESULT SECTION, since defining it for the lamp drive output port is sufficient. The two parameters are as follows:

- **sbo-name** is the name of the actual single break output port. This entity must be given a name for all conditions except: 1, an unused output; or 2, an output which flashes but has no steady state. The Compiler Program does not generate an equation for this parameter, it must be supplied. This is a major difference between the single break and lamp drive outputs.
- **sbo-flash** is the result of the equation which defines the conditions for flashing the output. This parameter is treated as a CURRENT RESULT type, but it does not need to be listed in the CURRENT RESULT SECTION. This parameter name is only needed if the output is required to flash.

In addition to the previous entries and the AOCD check value, parameters that represent the SBO 'on' and flash states are available. These parameters are only available for systems where Vital flashing is required, on boards which specify FLASH or state parameters; the 'on' state parameter is not available for restrictive flashing aspects. Their values represent the actual state of the output from the previous 1-second cycle, on, or flashing. The 'state' parameters cannot be used in their complemented state, that is, '.N.name' in an equation.

Example 5, all parameters, including the 'state' parameter:

1,sbo-name1 = -wire-name

1 (FLASH), sbo-flash1

1 (CK), sbo-ck1

1 (ON-STATE), sbo-on-state1

1 (FLASH-STATE), sbo-f1-state1

- sbo-ck is the name of the AOCD check parameter.
- sbo-on-state is the state of the output during the previous 1-second cycle. This parameter is true if the output was actually on steady for a full second. Not available for restrictive flashing aspects.
- sbo-fl-state is the state of the output during the previous 1-second cycle. This parameter is true if the output was actually flashing during that period.

The 'on-state' parameter is optional, and need only be defined if the user wishes to use the value in some equation. It is a special kind of parameter; neither current result nor self-latched, and the Compiler Program assigns this type of parameter accordingly. This state parameter is available whenever an SBO board description has a reference to a FLASH name or an on-state parameter name. There is no parameter for the OFF state since maintaining an output in its off state is a Vital function of the iVPI; therefore, an incorrect output for a false output parameter results in loss of power to all outputs. The false output parameter is a sufficient indication of an output's off state.

5.1.5.3.2 Data Records for SBO Board

Data records for SBO boards include the allowable options.

In the example, Ports 1 and 2 are assigned with their AOCD check parameters. Ports 3 and 4 are prewired spares, ports 5 and 6 have flash options, and ports 7 and 8 are assigned without AOCD check parameters or flash options. The first board is in slot 13, and is paired with the second board in slot 14. In the second board, only ports 1 and 2 are assigned. Three group wiring examples are shown.

For example:

```
SLOT 13 = SBO BOARD(14), 31166-430 GR 01
GROUP A                 = B12, N12-
1, 1G-SBO               = 1GR
1(CK), 1G-CK
2,2G-SBO                = 2GR
2(CK), 2G-CK
3,                                          = 3GR
4,                                          = 3VRELR
GROUP B                 = B12, N12-
5,EH-SBO = EH-REF
5(FLASH), EH-FLASH
6,ED-SBO = ED-REF
6(FLASH), ED-FLASH
6(ON-STATE), ED-ON
6(FLASH-STATE), ED-FL-ST
7,7G-SBO = 7GR
8,9VREL-SBO                 = 9VRELR
*
SLOT 14 = SBO BOARD(13), 31166-430 GR 01
GROUP A =B24, N24
1,12NW-SBO              = 12NWR
2,12RW-SBO              = 12RWR
```

5.1.5.4 Double Break Output Board

Double Break Output boards (DBO type) contain eight outputs divided into two groups of four outputs each. A GROUP data record is required to specify the positive and negative supplies for each group. GROUP A contains the wiring for ports 1 through 4, and GROUP B wiring supplies energy for ports 5 through 8.

DBO boards are logically grouped in pairs, via the slot number reference following the words 'DBO BOARD' on the data record. They can be paired in an address group with any other Vital output board type. In addition, there can be one unpaired Vital output board per module.

Each output port used requires an I/O port data record to specify the symbolic name of the output port and the wire names of the positive and negative wires that the port is attached to through the plug coupler.

If the logical value of the absence-of-current-detecting (AOCD) check circuit is to be used, another I/O port data record is required to provide the symbolic name of the AOCD circuit. For double break outputs, the second port data record includes the number of the port being referenced, and the option (CK) following the number. Each AOCD circuit to be used must then be assigned a unique name, and it must differ from the name of the output being checked.

The port numbers are assigned to the board from top to bottom, numbered 1 to 8.

NOTICE

If two boards are paired, the ports are still identified 1 to 8 for the first board and 1 to 8 for the second board.

Double break outputs can be driven steady or 'flashing'. The flash condition generates a code rate of on for 1/2 second and off for 1/2 second. If any Vital output in a system is required to flash, then all Vital outputs are susceptible to inadvertent flashing when they are supposed to be on steady. In many cases a flashing lamp drive output may be a more permissive aspect than a steady output, so there is a means to protect lamp drive outputs (see the Lamp Drive Output Board Application Rules for details).

The rules for assigning flash options to double break outputs are identical to the rules for single break output flash options.

5.1.5.4.1 Data Records for DBO Board

Data records for DBO boards include the allowable options.

In this example, ports 1 and 2 are assigned with their AOCD check parameters, ports 3 and 4 are prewired spares, ports 5 and 6 are unassigned, and ports 7 and 8 have flash options; GROUP A and GROUP B wiring is shown:

SLOT 01 = DBO BOARD (2), 31166-433 GR 01
GROUP A = B12-2, N12-2
1, 1NW-DBO = 1NW, 1RW
1(CK), 1NW-CK
2, 1RW-DBO = 1RW, 1NW
2(CK), 1RW-CK
3, = 3NW, 3RW
4, = 3RW, 3NW
GROUP B = B12-2, N12-2
7, EH-DBO = W-C2CNTL, W-CNTLREF
7(FLASH), EH-FLASH
7(ON-STATE), EH-DBO-ON
8 EBD-DBO = W-C3CNTL, W-CNTLREF
8(FLASH), EBD-FLASH
8(ON-STATE), EBD-DBO-ON
8(FLASH-STATE), EBD-DBO-FL

5.1.5.5 AC Output Board

The AC Output Board (ACO type) contains eight outputs, divided into two groups of four for power supply wiring. Each group of four requires a GROUP data record to specify the positive and negative (common) supplies for the four ports in each group.

ACO boards are logically grouped in pairs for addressing purposes (since 16 outputs can be addressed at a time). Boards can be paired via the slot number reference following the words 'ACO BOARD' on the slot assignment record. The paired boards must be in adjacent slots in the module, but paired boards for a single address do not have to be the same Vital output board type. There can be one unpaired Vital output board per module.

Each output port used requires an I/O port data record to specify the integer identification of the output port. The numbers are assigned from top to bottom on the board.

The data record must also include the name of the positive wire that the port is attached to through the plug coupler. Each output port requires the specification of the positive only, since the four output ports in a group are referenced to the negative supply specified on the GROUP data record for the four. Two plug coupler pins are supplied for each output, but one side is always referenced to the group negative energy and can be used as a tie point, if necessary.

If the logical value of the absence-of-current-detecting (AOCD) check circuit is to be used, another I/O port data record is required to provide the symbolic name of the AOCD circuit. The second port data record includes the number of the port being referenced, and the option (CK) following the number. Each AOCD check value used must be assigned a unique name and it must differ from the name of the output being checked. This (CK) value is used to detect the presence or absence of load current on an output.

NOTICE

Available in iVPI CAA 611a and later.

Unlike the other Vital output board types, the logical value of the absence-of-current-detecting (AOCD) check circuit of an ACO cannot be determined over an entire 1 second cycle. ACOs cannot be flashed except when restrictive flashing aspects are selected, in which case AOCD data for the flash-state parameter is collected only during the last half of the cycle. The non-vital “output check test” for an ACO board is set logic TRUE by the system when the current flow is detected to be above the AOCD level of the output; it is set FALSE by the system when the current flow is below the AOCD level. Flash-only ports can be specified by using the wire name only for the output port and entering a FLASH parameter.

WARNING

ACO CURRENT CHECK PARAMETER APPLICATION

The ACO current check parameter is calculated in a non-vital manner and shall not be applied as a fail-safe parameter.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

For example:

1, = wire-name

1 (FLASH), aco-flash-name

The port numbers are assigned to the board from top to bottom, numbered 1 to 8.

NOTICE

If two boards are paired, the ports are still identified 1 to 8 for the first board and 1 to 8 for the second board.

5.1.5.5.1 Data Records for ACO Board

Data records for ACO boards include the allowable options.

For example:

```
SLOT 4 = ACO BOARD(0), 31166-432 GR 02
GROUP A =          B12,          N12-
1, 2EAG =          N2EAGE
1 (CK), 2EAG-CK
2, 2EAR =          N2EARE
2 (CK), 2EAR-CK
3, 2EAY =          N2EAYE
3 (FLASH), N2EAYE - FL
3 (FLASH-STATE), N2EAYE-FL-ST
4,         =                    13GR
4 (FLASH), 13GR-FL
GROUP B =          B12,          N12-
5,         =                    13GH
```

In this example, ports 1 and 2 are assigned with their AOCD check parameters, ports 3 and 4 are assigned, port 5 is a prewired spare, and ports 6, 7, 8 are unused. The ACO board is in slot 4 and it is unpaired. The flash parameters for ports 3 and 4 are allowed only for restrictive flashing. Port 4 is flash-only.

5.1.5.6 Restrictive Flashing Aspects

Restrictive flashing refers to applications where output flashing is a more restrictive condition than steady on. When restrictive flashing is in effect, outputs can be protected in two ways:

1. For any output port capable of flashing, a FLASH-STATE parameter can be defined for use in the equations. This parameter is true only if no current flowed in that output during the last half of the previous 1-second cycle. This condition includes output flashing, off or inoperative.
2. If RESTRICTIVE FLASHING ASPECTS = CHECKWORD is selected, the output checkword is incorrectly formed if an output should be flashing but is on steady; the VRD drops. Flash state parameters are still available in this mode.

The CAA does not automatically generate equations for restrictive flash protection; these must be created by the user.

5.1.6 NVSP Boards

Up to four NVSP boards can exist in an iVPI system. NVSP board must be placed within slots 2-8 of the module.

5.1.6.1 Software Revision / Site ID

A 6-bit software revision ID must be specified for each NVSP board. This value is selected using switches on the board and programmed into the NVSP application so that, if the selected value does not match the programmed value, the NVSP application does not run. To program the expected revision ID value, enter a SOFTWARE REVISION ID record in the .CSI file.

The NVSP board can also optionally be set up so that the NVSP application only runs if the 10-bit site ID selected at the back of the VSP board matches an expected value programmed into the NVSP application. To program the expected site ID value, use the VSP SOFTWARE SITE ID or the VSP SYSTEM ID record in the .CSI file

5.1.7 Genrakode Track Processor (GTP) Boards

A total of 8 GTP boards are allowed for an entire system; each board requires two slots. A total of up to ten boards of Vital Serial type (GTP and CRG) are allowed in the system.

The VSP board communicates with GTP boards through DPRAM using a type of Vital Serial message: see the section on GTP Communications for more details.

Each GTP board contains an application and system software programmed by a tool separate from CAAPE – the GTP CAA. Certain shared information such as message Link and Block assignments can be exported from the iVPI Compiler and imported by the GTP CAA. GTP EXPORT FILE records in the iVPI Compiler's .VPC file can be used to specify the base names of the export files for the GTP boards in a system. The iVPI compiler adds an extension of iGT to each of the files it creates.

5.1.8 Code Rate Generator (CRG) Boards

A total of 3 CRG boards are allowed for an entire system. A total of up to ten boards of Vital Serial type (GTP and CRG) are allowed in the entire system. iVPI CRG boards have two groups of four ports each.

The VSP board communicates with CRG boards through DPRAM using a type of Vital Serial message; see the section on CRG Communications for more details.

Each Code Rate Generator board has a prewired identifier determined by the module and slot in which it is located. This identifier is used to make each CRG board unique so that it cannot accidentally use data meant for another board in the system. Limitations in iVPI wiring cause some of the available CRG identifiers to be repeated when there is more than one module. CRG boards cannot be placed such that more than one board has the same ID. The Compiler enforces this restriction so that applications cannot be built if CRG boards have duplicate ID values.

Use Table 5–4 when placing CRG boards to ensure that no CRG board identifier is duplicated within a system.

Table 5–4. CRG Board ID Assignments by Slot

Slot	System Module CRG ID	Expansion Module 1 CRG ID	Expansion Module 2 CRG ID	Expansion Module 3 CRG ID
2	EE	CE	EE	CE
3	AC	8C	AC	8C
4	6C	4C	6C	4C
5	2B	0B	2B	0B
6	EB	CB	EB	CB
7	A9	89	A9	89
8	69	49	69	49
9	36	16	36	16
10	F6	D6	F6	D6
11	BE	9E	BE	9E
12	7E	5E	7E	5E
13	3D	1D	3D	1D
14	FD	DD	FD	DD
15	BB	9B	FB	9B
16	EA	CA	AA	4A
17	A8	88	68	08
18	68	48	28	C8
19	27	07	E7	87
20	E7	C7	A7	47
21	8F	85	4F	05

5.1.9 Non-Vital Input (NVI) Boards

Each Non-Vital Input (NVI) board contains 32 differential input ports, each port requiring a decimal integer port identification and wiring assignments for the positive and negative wires.

There can be multiple NVSP boards in a module, for instance, one board performing code system emulation and a second controlling the non-vital I/O. The non-vital I/O must be assigned to a NVSP board using the ID number assigned to the NVSP board on the slot assignment record.

Input port numbers are assigned to the board from top to bottom, numbered 1 to 32.

The inputs are split into four groups of eight, and each eight are then referenced to common positive and negative energy via the GROUP data record. There are four GROUPS per board, GROUP A through GROUP D. Each group of eight can be assigned one of the three allowable debounce values: 0, 25 or 50 milliseconds. To specify one of these options, the DEBOUNCE record is used.

NVI boards have the Use Diagnostics option selected and grayed out to prevent the user from disabling Diagnostics.

5.1.10 Non-Vital Output (NVO) Boards

Each Non-Vital Output board contains 32 outputs. The output port numbers are assigned to the board from top to bottom, numbered 1 to 32.

Outputs are split into four groups of eight, and each group of eight is connected to a common reference via the GROUP data record. There are four GROUPS per board, GROUP A through GROUP D.

For boards that require external power supplies, a PS A record must be used to designate the power supply wiring for any ports in Groups A and B and a PS B record must be designated for any ports in Groups C and D.

NVO RELAY boards have connections for Form-C relays in Groups A and C, with front, back and heel wires; Groups B and D have connections for Form-A relays, with front and heel wires only. NVO SSR boards have front and heel wires only. NO WIRE can be used to designate unwired connections.

NVO boards have the Use Diagnostics option selected and grayed out to prevent the user from disabling Diagnostics. Output pulse and flash options are available. Default flash rate is 60 per minute (that is, on for 1/2 second, off for 1/2 second) for normal flash and 120 per minute (1/4 second on, 1/4 second off) for fast flash. Flashing can occur in two phases, such that for two outputs flashing at the same rate but different phase, one is on when the other is off. Two pulse widths are available: defaults are 200 milliseconds for PULSE and 400 for PULSE2. Default values can be changed using hardware options.

Output pulsing or flashing is enabled only when an output is on (true). If pulse or flash modes are selected, the output state is determined by the following order of precedence.

Table 5–5. Non-vital Output States

Output	Mode	State
FALSE	any	off
TRUE	none	on steady
TRUE	FLASH2B	fast flash, phase B
TRUE	FLASHB	normal flash, phase B
TRUE	FLASH2	fast flash, phase A
TRUE	FLASH	normal flash, phase A
TRUE	PULSE	pulse, width 1
TRUE	PULSE	pulse, width 2

When PULSE2 or PULSE is enabled, a single pulse of the appropriate width is delivered. The output must be turned off to reset the port before another pulse can be delivered.

5.2 VSP-NVSP COMMUNICATIONS

VSP-NVSP communications are the DPRAM communications between the Vital application on the VSP board and the non-vital applications on the NVSP boards in the system.

Communications between the VSP and each NVSP board consist of application variables passed between the boards once per VSP system cycle. For each NVSP board in a system, two messages can be defined by the user: one VSP-to-NVSP message that is sent from the VSP to the NVSP, and one NVSP-to-VSP message that is sent from the NVSP to the VSP. In addition, a diagnostic message from NVSP to VSP is automatically created by the compiler.

It is acceptable for NVSP boards to have no user-defined messages; however, the diagnostic message is still created.

5.2.1 VSP-to-NVSP Messages

This message defines the Vital application variables passed from the VSP board to the NVSP board. A single message is typically defined and shared between the Vital and non-vital applications; this means that the message variable names are the same in both applications.

In the Vital application, any variable except timer variables can be sent to the NVSP board. In the non-vital application, VSP-to-NVSP message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a place-holder for unused message bits.

5.2.2 NVSP-to-VSP Messages

This message defines the non-vital application variables passed from the NVSP board to the VSP board. A single message is typically created and shared between the Vital and non-vital applications; this means that the message variable names are the same in both applications.

In the non-vital application, any Boolean variable or array element can be sent to the VSP board. In the Vital application, NVSP-to-VSP message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a place-holder for unused message bits.

5.2.3 NVSP-to-VSP Diagnostic Messages

This message is automatically created by the compiler. It includes a single usable variable.

CSEX*n*-ALIVE

- *n* is the NVSP board number:
  - CSEX1-ALIVE for NVSP board number 1
  - CSEX2-ALIVE for board 2, etc.

5.3 GTP COMMUNICATIONS

GTP communications are the DPRAM communications between the Vital application on the VSP board and the applications on the GTP boards in the system. GTP boards contain an application that is programmed by a tool separate from CAAPE.

One VSP-to-GTP and one GTP-to-VSP message is defined per GTP board. Message lengths are fixed for a given version of GTP software. Each message is subdivided into two sections, one per track. The meaning of specific message bits depends on the version of GTP software and the programming of the GTP board. A GTP board diagnostic message is automatically created by the compiler for each GTP board.

5.3.1 VSP-to-GTP Messages

This message is sent from VSP to a GTP board. Tracks are designated A or B; each track is assigned a number of bits based on the GTP software version. Any variable type except timer variables can be included in the message. PERMZERO can be used as a place-holder for unused message bits.

5.3.2 GTP-to-VSP Messages

This message is sent from a GTP board to the VSP board. It is partitioned the same as the VSP-to-GTP message. GTP-to-VSP message variables are considered application inputs to VSP and their names cannot be declared as other variable types. PERMZERO can be used as a place-holder for unused message bits.

5.3.3 GTP Diagnostic Messages

This message consists of a single variable called GTPn-ALIVE.

5.3.3.1 GTP Links and Blocks

GTP boards use sets of codeword values to ensure Vital communications between VSP and GTP. These codeword sets must be consistent on both sides of the link between communicating boards, and must not be duplicated within a system. The CAA compiler enforces uniqueness of codeword sets within the iVPI system.

⚠WARNING

VITAL COMMUNICATIONS REQUIRE UNIQUE LINK AND BLOCK SETTINGS

Failure to properly assign, maintain and control unique Link and Block settings for Vital communications within iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

The message link and block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link and Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

See SECTION 5.8 – Vital Serial Links and Blocks for more information.

5.4 CRG COMMUNICATIONS

CRG communications are the DPRAM communications between the Vital application on the VSP board and each of the CRG boards in the system.

CRG communications consists of one data message sent to each CRG board and one status message received from each CRG board in the system. A CRG board diagnostic message is automatically created by the compiler for each CRG board.

5.4.1 CRG Data Messages

The data message sent to the CRG board consists of subsections, one for each port on the board. CRG ports are numbered 1 through 8. Each subsection has 10 message bits. Any variable except for timer variables can be used as a message bit. PERMZERO can be used as a place-holder for unused message bits.

5.4.2 CRG Status Messages

Status data received from a CRG board consists of one bit per port. Message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a place-holder for unused message bits.

⚠WARNING

CRG STATUS PARAMETER APPLICATION

The CRG status parameters are calculated in a non-vital manner and must not be applied as fail-safe parameters.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

5.4.3 CRG Diagnostic Messages

This message is automatically created by the compiler to provide diagnostic data on the CRG board.

NOTICE

At this time, it is recommended that this parameter NOT be used as it is not fully functional/supported and results could yield errors.

5.4.3.1 CRG Links and Blocks

For CRG, the Link and Block numbers are not defined by the user, but are automatically assigned by the compiler. See SECTION 5.8 – Vital Serial Links and Blocks for more information on link and block numbers.

5.5 NETWORKING

VSP and NVSP boards are equipped with Ethernet controller devices and capable of network communications when running the appropriate system software. These boards have two network devices, designated ENET1 and ENET2; either or both of these devices can be configured for network communications.

These types of network communications are available:

- MAC TCP: VSP and NVSP communications with the Alstom Maintenance Management System (MMS).
- Vital Serial over Ethernet (VSoE): VSP Vital data communications based on Vital Serial algorithms.
- DigiSAFE® Vital communications: VSP Vital data communications based on DigiSAFE Vital Serial algorithms.

NOTICE

DigiSAFE communications is only available in certain versions of the Vital CAA.

- NonVital Serial over Ethernet (NVSoE): NVSP non-vital data communications using various code system emulations and other protocols.

5.5.1 IP Addresses and Subnets

The boards in iVPI use fixed IP addressing where the location of each network device is specified by a fixed *IP address* given in "dotted quad" notation such as 121.80.5.3. Dynamic addressing such as DHCP is not currently available.

Subnets are logical subdivisions that partition a network into sections. They might be used to restrict access between one group of network devices and another, or to reduce overall message traffic. The addresses of all the devices on a subnet include a part which they share in common; this shared subnet address can also be represented in dotted quad notation such as 121.80.5.0. A subnet mask is used to specify the size of the shared address area common to all the devices on the subnet and also implies the maximum number of devices a subnet can contain.

The default subnet mask used in CAAPE is 255.255.255.0. With this mask, the first three digits of an IP address identify a subnet and the last digit can vary to distinguish the individual devices on the subnet. For example, 121.80.5.3, 121.80.5.4 and 121.80.5.5 would all be device addresses in subnet 121.80.5.0; 122.81.6.10, 122.81.6.15 and 122.81.6.99 would all be device addresses in subnet 122.81.6.0. The last address digit can vary from 1 to 254, allowing up to 254 devices on a subnet.

The network devices on a given board must be on separate subnets, but the devices on different boards can be on the same subnet. For example, the ENET1 devices of a VSP and a NVSP board could both be on one subnet and their ENET2 devices on another. The combination of a device's IP address and a subnet mask identifies the subnet on which the device resides. Devices cannot communicate directly across subnets; they must go through one or more gateways, which are devices for bridging subnets. If a local device has to communicate with a remote device on a different subnet, the IP address of a gateway must be specified. The gateway address must be on the same subnet as the local device.

Older iVPI software and CAA versions prior to iVPICAA 610 did not allow for routing (communications across subnets). A fixed subnet mask of 255.255.255.0 was used in all cases.

The details of calculating subnet masks and allocating devices to subnets is outside the scope of this text. Tools are available for doing subnet calculations.

5.5.2 Redundancy

Newer iVPI software and newer CAA versions (610 and later) allow network links that are path-redundant. Identical copies of every message are sent though both network devices to a remote node; the messages contain information that allows the recipient to identify and discard duplicate messages so that ultimately only one of the copies will be used. Message data thus takes two separate network paths to reach the same recipient. If one path fails, the data can still reach the recipient over the other path.

5.5.3 Client / Server

For some types of communications the local and remote devices have a client / server relationship. The term can have two different but related meanings in iVPI.

Functionally, a server is a program that provides specific services to one or more client programs; a server typically waits for clients to initiate communications with it and request its services.

In addition, certain types of communications that are based on TCP/IP (Transmission Control Protocol/Internet Protocol is connection oriented) rather than UDP (User Datagram Protocol is a connectionless Internet Protocol) require that, when two nodes are to be connected, one of the nodes must be designated as the one that will initiate the request to establish the connection. The node that initiates the connection request is called the client and the node that passively waits for the request to be made is called the server. Thus for certain protocols such as DT8 PEER the nodes do not inherently have client / server functionality but must still be designated as client or server so they can correctly establish their network connection.

5.5.4 Ethernet Ports

A given physical network device may be used for various types of communications. For example, a VSP board may have MACTCP communications with a Maintenance Management System (MMS) as well as Vital Serial over Ethernet communications with remote iVPI VSP boards. The different types of communications are implemented as separate software processes running simultaneously on the board. A way must be provided by which messages are directed, not only to the receiving network device on the target board, but also to the correct software process that will handle it.

Ethernet communications uses the concept of a "socket" in describing how data is passed between software processes on different network devices. A network link can be considered to have a socket at each end. Each socket is defined by the IP address of the network device plus a numeric "port value" which identifies the software process that will handle the messages. The rules for assigning port values vary depending on the type of communications and the protocol it is using. CAA packages will assign the appropriate port values automatically where possible, but the user may manually enter port values where demanded by the protocol or to override automatic assignment for special cases. CAA versions 610 and later now assign port values from a range of general-usage, non-registered values, but will also provide ways to continue to use the old range of values when necessary for communications with existing applications.

See the sections on VSP and NVSP networking for details on Ethernet port value assignments.

5.6 NETWORK COMMUNICATIONS (VSP)

5.6.1 Introduction

VSP boards have two network devices, designated ENET1 and ENET2, and a separate processor for controlling network communications. Either or both devices can be used.

Network communications is enabled on a VSP board by adding the keyword ETHERNET to the board's slot record in the hardware file:

```
SLOT 1 = VSP BOARD, 31166-427 GR01, ETHERNET
```

When network communications is enabled, the Comm processor performs network operations. The Main processor performs Vital operations such as vital I/O and vital logic, and interfaces to the Comm processor for access to the network. The Main and Comm processors each have separate applications that are prepared by different compilers. Having separate Main and Comm applications allows network settings to be changed without having to recompile and retest the entire Main application.

Each network device that is used must be assigned a different IP address.

5.6.2 Network-Based Diagnostics

Alstom's MMS tool can be used to gather system diagnostics over the network. The user must enable MMS diagnostics on the network device(s) to which MMS will connect. This is done in one of two ways.

For CAA versions prior to 610, add the DIAGNOSTICS field to the ENET1 DEVICE or the ENET2 DEVICE compiler input record. The CAA will not prevent MMS diagnostics from being enabled for both devices, but this configuration may have unpredictable effects on system operation and is not recommended.

```
ENET1 DEVICE = IP(172.13.15.19), DIAGNOSTICS(YES)
```

For CAA versions 610 and later with subnets and redundancy, specify the network device in a MACTCP compiler input record: ENET1, ENET2 or REDUNDANT for a path-redundant MMS link using both devices. The specified device(s) must have defined IP addresses.

```
MACTCP = DEVICE(ENET1), ...
```

Ethernet socket port values are assigned as follows:

- CAA versions before 610: 1100 for ENET1, 1101 for ENET2
- CAA versions 610 and later with subnets/redundancy: 51100 for ENET1, 52100 for ENET2

The user must generate an MMS data file containing the information needed for MMS to connect to the system and display diagnostics. See the CAAPE User's Guide for information on generating MMS files. The MMS file will contain the CAA-assigned Ethernet socket port values, so there is generally no need for the user to know these values.

5.6.3 Vital Serial Over Ethernet

Vital Serial Over Ethernet (VSOE) is a type of Vital Serial Communications performed over a network. Central to VSOE is the concept of a *VSOE Node*. Nodes can be considered virtual Vital Serial boards; each node can send to and/or receive from a remote unit in another system. VSOE messages for a given node are sent and received through one or both of the network devices depending on whether path-redundancy is used; nodes can be assigned to their respective devices through compiler input records.

5.6.3.1 VSOE Node Types

The node's type designates how it behaves in the network. Available types are:

- PEER – bidirectional communications with another VSOE node

5.6.3.2 Output VSOE Messages

This message sends application variables through the source VSOE node. Any application variables except for timer variables are allowed. A maximum message length of 450 variables for VSOE is allowed. PERMZERO can be used as a place-holder for unused message bits.

5.6.3.3 Input VSOE Messages

This message defines the variables received from a remote VSOE message. This is the same physical message output by the remote VSOE node, although different variable names might be used on either side of the link for consistency within each application. For example, the Vital input VRDFRNT-DI might be output from one VSOE but be called something like REMOTE-VRDFRNT in the receiving application. Input VSOE message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a placeholder for unused message bits. A maximum message length of 450 variables for VSOE is allowed.

5.6.3.4 VSOE LINKOK

This variable is not intended for use and is not supported at this time.

5.6.3.5 IP Addresses and Network Ports

A sending VSOE node must know the local network device(s) from which it must send the message, and how to connect to the remote nodes. A receiving VSOE node must know what node has sent a particular message so that its data can be routed to the corresponding VSOE Input Message in the Vital application. Two elements identify an end point of a network communications channel: an IP address and an Ethernet socket port number. CPU II network devices are assigned an IP address by the user; rather than forcing the user to manage Ethernet port numbers, the CAA assigns default VSOE port numbers to each device and the user only has to identify which remote network device is to receive the message. Therefore:

- For VSOE-to-VSOE transmit: the user specifies the local network device from which to transmit, the remote IP address, and the remote network device which implies a default CAA-assigned Ethernet port number. The remote IP address and port number are used to open a channel to the remote node and send the message.
- For VSOE-to-VSOE receive: the user specifies the local network device where the message will be received, the IP address of the remote transmitting node, and the remote network device which implies a default CAA-assigned Ethernet port number. When a message is received, the IP address and port number of the sender is used to identify which remote node sent the message and determine how to route the data to the Vital application.

CAA versions 610 and later with subnets/redundancy now assign Ethernet port values from a range of general-usage, non-registered values. However, where a new application must communicate with an existing one created using aCAA version before 610, the user can specify that the old Ethernet port assignment scheme is to be used.

CAA-assigned Ethernet port assignment values are:

- CAA versions before 610: 1102 for ENET1, 1103 for ENET2
- CAA versions 610 and later with subnets/redundancy: 51101 for VSOE ENET1, 52101 for VSOE ENET2

5.6.3.6 Link and Block Numbers

Link and block numbers are used in messages between VSOE nodes to ensure that the message codewords are known on both sides of the link and to prevent a message sent to the wrong recipient from being decoded to give permissive values.

⚠WARNING

VITAL COMMUNICATIONS REQUIRE UNIQUE LINK AND BLOCK SETTINGS

Failure to properly assign, maintain and control unique Link and Block settings for Vital communications within iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

The message link and block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link and Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

See SECTION 5.8 – Vital Serial Links and Blocks for more information.

5.6.3.7 Coordination of VSOE Input Files

Several files are used to specify data for VSOE:

- .VNT file: VSOE node declarations for the local system, shared by Vital and Vital Comm compilers.
- .CW file: network links and vital link data, shared by Vital and Vital Comm compilers; could be shared across entire network.
- .VSL file: specifies VSOE messages for the local system, used by Vital compiler only.
- .VCC file: properties of the network devices for the local VSP board, used by Vital Comm compiler only.
- .NVS file: IP address and port information on local and remote VSOE nodes, used by Vital Comm compiler only, could be shared across entire network.
- .GW file: IP addresses of gateways used to access remote nodes, used by Vital Comm compiler only; could be shared across entire network.

Records in the .VNT file identify the names and types of the VSOE nodes in the system being compiled. Node names are used to identify the nodes in subsequent files. A sequential number from 1 to the number of nodes in the system identifies the node when data is passed between the Vital and the Vital Comm applications, e.g. the fifth node in the Comm processor's list corresponds to the fifth node in the Main processor's list. In the following example, there are two PEER VSOE nodes.

VSOE 1 ID = TYPE(PEER), BA4

VSOE 2 ID = TYPE(PEER), BA5

The order of the nodes in the .VNT text file reflects the message transmission order from the vital processor board; VSOE node 1 transmits first during the system cycle. VSOE communications operate most efficiently based on the following order based on application size determined by the number of equations and VSOE nodes (a large application contains more logic equations or VSOE nodes than a smaller application):

1. Messages to other VPI/iVPI nodes that contain large applications
2. Messages to other VPI/iVPI nodes that contain medium applications
3. Messages to other VPI/iVPI nodes that contain small applications
4. AFCPU2 and microWIU nodes

Records in the .CW file identify the attributes of a send or receive link; SOURCE and DESTINATION records identify the senders and receivers of a message. In the following example, the PEER VSOE has both transmit and receive links. Communications from local node BA4 to remote node ABC is done through gateway “GW-1” whose IP address will be given in a separate file.

```
VSOE LINK 1 = LENGTH(50),BLOCK(5)
SOURCE = BA4
DESTINATION = ABC, GATEWAY(GW-1)

VSOE LINK 2 = LENGTH(55),BLOCK(6)
SOURCE = ABC
DESTINATION = BA4
```

Records in the .VSL file identify the contents of the message. In the example below, the PEER VSOE has transmit and receive messages.

```
SOURCE = BA4
1 = VRDFRNT-DI
 …
DESTINATION = BA4
1 = REMOTE-VRDFRNT
…
```

The ENET1 DEVICE and ENET2 DEVICE records in the .VCC file give the IP addresses (and subnet masks, for CAA versions that support subnets) of the network devices.

```
ENET1 DEVICE = IP(172.11.23.34), MASK(255.255.255.0)
```

Records in the .NVS file identify the network properties of the nodes. The nodes used by a given system are identified by matching their NAME fields to the names in the .VNT file for local nodes, or to names in the .CW file for remote nodes. In the following example, network properties are given for the nodes described above. The Comm compiler recognizes that BA4 and BA5 are local nodes because they match names in the .VNT file; their IP addresses match the IP address of ENET1 from the .VCC file, so the compiler recognizes that ENET1 is the source device from which messages are transmitted and received. ABC, OS1 and OS2 are recognized as remote nodes because their names match the names of remote sources and destinations in the .CW file. Their IP address and port information are used to connect to the remote nodes.

```
VSOE = NAME(BA4),IP(172.11.23.34),PORT(ENET1)
VSOE = NAME(BA5),IP(172.11.23.34),PORT(ENET1)
VSOE = NAME(ABC),IP(172.11.24.35),PORT(ENET2)
VSOE = NAME(OS1),IP(172.11.23.36),PORT(ENET1)
VSOE = NAME(OS2),IP(172.11.23.37),PORT(ENET1)
```

Records in the .GW file identify the network properties of any gateways used in the network. The gateways used by a given system are identified by matching their NAME fields to the names in the source files where gateways are used.

```
GATEWAY = NAME(GW-1), IP(172.11.23.45)
```

5.6.4 DigiSAFE

NOTICE

DigiSAFE Communications is available in only certain iVPI CAAs.

DigiSAFE is a type of Vital Serial Communications performed over a network. Central to DigiSAFE is the concept of a *DigiSAFE Node*. Nodes can be considered virtual Vital Serial boards; each node can send to and receive from a remote unit in another system. DigiSAFE messages for a given node are sent and received through both of the network devices. Path-redundancy must be used; nodes can be assigned to their respective devices through compiler input records.

5.6.4.1 DigiSAFE Node Types

The node's type designates how it behaves in the network. The only available type is:

- DS_PEER – bidirectional communications with another DigiSAFE node

5.6.4.2 Output DigiSAFE Messages

This message sends application variables through the source DigiSAFE node to one or more remote DigiSAFE nodes. Any application variables except for timer variables are allowed. A maximum message length of 160 variables for DigiSAFE is allowed. PERMZERO can be used as a placeholder for unused message bits.

5.6.4.3 Input DigiSAFE Messages

This message defines the variables received from a remote DigiSAFE message. This is the same physical message output by the remote DigiSAFE node, although different variable names might be used on either side of the link for consistency within each application. Input DigiSAFE message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a placeholder for unused message bits.

5.6.4.4 IP Addresses and Network Ports

A sending DigiSAFE node must know the local network device(s) from which it must send the message, and how to connect to the remote nodes. A receiving DigiSAFE node must know what node has sent a particular message so that its data can be routed to the corresponding DigiSAFE Input Message in the Vital application. Two elements identify an end point of a network communications channel: an IP address and an Ethernet socket port number. VSP network devices are assigned an IP address by the user and a DigiSAFE Ethernet port number by the user. Therefore:

- For DigiSAFE-to-DigiSAFE transmit: the user specifies the local network devices from which to transmit, the remote IP addresses, and the remote Ethernet port number. The remote IP addresses and port number are used to open a channel to the remote node and send the message.
- For DigiSAFE-to-DigiSAFE receive: the user specifies the local network devices where the message will be received, the IP address of the remote transmitting node, and the remote Ethernet port number. When a message is received, the IP addresses of the sender are used to identify which remote node sent the message and determine how to route the data to the Vital application.
- The default Ethernet port numbers for DigiSAFE communications are 61440 for ENET1 and 61440 for ENET2.

5.6.4.5 Coordination of DigiSAFE Input Files

Several files are used to specify data for DigiSAFE:

- .VPC file: Contains the iVPI ID to identify a unique node on the DigiSAFE message network; used by Vital compiler only.
- .VNT file: DigiSAFE node declarations for the local system, shared by Vital and Vital Comm compilers.
- .CW file: network links and vital link data, shared by Vital and Vital Comm compilers; could be shared across entire network.
- .VSL file: specifies DigiSAFE messages for the local system, used by Vital compiler only.
- .VCC file: properties of the network devices for the local VSP board, used by Vital Comm compiler only.
- .NVS file: IP address and port information on local and remote DigiSAFE nodes, used by Vital Comm compiler only, could be shared across entire network.
- .GW file: IP addresses of gateways used to access remote nodes, used by Vital Comm compiler only; could be shared across entire network.

The iVPI ID record in the .VPC file identifies the iVPI on the network among all of the other DigiSAFE messages on the network. This number must be assigned a unique ID. The data type is a 16-bit unsigned integer. In the following example, an iVPI ID is defined:

iVPI ID = 33452

⚠WARNING

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE VSP BOARD ID

A unique ID must be assigned to each iVPI VSP board in order to give each iVPI a unique identification in the network of DigiSAFE communications.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

Records in the .VNT file identify the names and types of the DigiSAFE nodes in the system being compiled. Node names are used to identify the nodes in subsequent files. A sequential number from 1 to 3 nodes in the system identifies the node when data is passed between the Vital and the Vital Comm applications, e.g., the second node in the Comm processor's list corresponds to the second node in the Main processor's list. In the following example, there are two DS_PEER DigiSAFE nodes.

DigiSAFE 1 ID = TYPE(DS_PEER), DS_01, ZCID=1234

DigiSAFE 2 ID = TYPE(DS_PEER), DS_02, ZCID=3456

⚠WARNING

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE ZONE CONTROLLER ID

A unique ID must be assigned to each zone controller in order to give each zone controller a unique identification in the network of DigiSAFE communications.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

DIGISAFE COMMUNICATIONS REQUIRE UNIQUE NETWORK IDS

The DigiSAFE iVPI IDs and the zone controller IDs must be assigned such that the values are unique to each other and to any other ID throughout the entire DigiSAFE network.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

Records in the .CW file identify the attributes of a send or receive link; SOURCE and DESTINATION records identify the senders and receivers of a message. In the following example, the DS_PEER DigiSAFE has both transmit and receive links. Communications from local node DS_01 to remote nodes REM_ZC_01A and REM_ZC_01B is done through gateway "GW-1" whose IP address will be given in a separate file. VSP, with the DigiSAFE protocol, communicates in a redundant network configuration to a redundant remote node.

DIGISAFE MSG 201 = LENGTH(160),REDUNDANT

SOURCE = REM_ZC_01A

SOURCE = REM_ZC_01B

DESTINATION = DS_01

DIGISAFE MSG 202 = LENGTH(160),REDUNDANT

SOURCE = DS_01

DESTINATION = REM_ZC_01A, GATEWAY(GW-01,GW-01)

DESTINATION = REM_ZC_01B, GATEWAY(GW-01,GW-01)

Records in the .VSL file identify the contents of the message. In the example below, the PEER DigiSAFE has transmit and receive messages.

SOURCE = DS_01

1 = VRDFRNT-DI

…

…

160 = PERMZERO

DESTINATION = DS_01

1 = 4NWZ

…

…

160 = PERMZERO

The ENET1 DEVICE and ENET2 DEVICE records in the .VCC file give the IP addresses (and subnet masks, for CAA versions that support subnets) of the network devices.

```
ENET1 DEVICE = IP(172.11.23.34), MASK(255.255.255.0)
```

Records in the .NVS file identify the network properties of the nodes. The nodes used by a given system are identified by matching their NAME fields to the names in the .VNT file for local nodes, or to names in the .CW file for remote nodes. In the following example network properties are given for the nodes described above. The Comm compiler recognizes that DS_01 and DS_02 are local nodes because they match names in the .VNT file; their IP addresses match the IP addresses of ENET1 and ENET2 from the .VCC file, so the compiler recognizes that ENET1 and ENET2 are the source devices from which messages are transmitted and received. REM_ZC_01A, REM_ZC_01B, REM_ZC_02A and REM_ZC_02B are recognized as remote nodes because their names match the names of remote sources and destinations in the .CW file. Their IP address and port information are used to connect to the remote nodes.

```
DIGISAFE = NAME(DS_01),REDUNDANT,IP(120.80.55.1,120.80.55.2),
    PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(REM_ZC_01A),REDUNDANT,IP(120.80.55.6,120.80.55.10),
    PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(REM_ZC_01B),REDUNDANT,IP(120.80.55.7,120.80.55.9),
    PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(DS_02),REDUNDANT,IP(120.80.55.1,120.80.55.2),
    PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(REM_ZC_02A),REDUNDANT,IP(120.80.55.4,120.80.55.5),
    PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(REM_ZC_02B),REDUNDANT,IP(120.80.55.8,120.80.55.11),
    PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
```

Records in the .GW file identify the network properties of any gateways used in the network. The gateways used by a given system are identified by matching their NAME fields to the names in the source files where gateways are used.

```
GATEWAY = NAME(GW-1), IP(172.11.23.45)
```

5.7 NETWORK COMMUNICATIONS (NVSP)

5.7.1 Introduction

Network-capable NVSP boards have two network devices, designated ENET1 and ENET2, and a separate processor for controlling network communications. Either or both devices can be used.

Network communications is enabled on an NVSP board by adding the keyword ETHERNET to the board's slot record in the hardware file:

```
SLOT 1 = NVSP BOARD, 31166-428 GR01, ETHERNET
```

When network communications is enabled, the Comm processor performs network operations. The Main processor interfaces to the Comm processor for access to the network. The Comm processor has no application; it processes commands sent to it by the Main processor, which implements the specific network communications protocols.

Each network device that is used must be assigned a different IP address.

5.7.2 Network-Based Diagnostics

Alstom's Maintenance Management System (MMS) tool can be used to gather system diagnostics over the network. The user must enable MMS diagnostics on the network device(s) to which MMS will connect. This is done in one of two ways.

For CAA versions prior to 610, add the DIAGNOSTICS field to the ENET1 DEVICE or the ENET2 DEVICE compiler input record. The CAA will not prevent MMS diagnostics from being enabled for both devices, but this configuration may have unpredictable effects on system operation and is not recommended.

```
ENET1 DEVICE = IP(172.13.15.19), DIAGNOSTICS(YES)
```

For CAA versions 610 and later with subnets/redundancy, specify the network device in a MACTCP compiler input record: ENET1, ENET2 or REDUNDANT for a path-redundant MMS link using both devices. The specified device(s) must have defined IP addresses.

```
MACTCP = DEVICE(ENET1), ...
```

Ethernet socket port values are assigned as follows:

- CAA versions before 610: 1100 for ENET1, 1101 for ENET2
- CAA versions 610 and later with subnets/redundancy: 51100 for ENET1, 52100 for ENET2

The user must generate an MMS data file containing the information needed for MMS to connect to the system and display diagnostics. See the CAAPE User's Guide for information on generating MMS files. The MMS file will contain the CAA-assigned Ethernet socket port values, so there is generally no need for the user to know these values.

5.7.3 Network Serial Ports (NVSOE)

Network serial communications is similar to standard non-vital serial communications, except that instead of a physical serial port the application uses a logical network port and NVSOE (NonVital Serial over Ethernet) node. A network serial port in the Main Processor consists of a set of related control, indication, text and special messages and a protocol for sending and receiving them; a corresponding NVSOE node in the Comm Processor performs the network communications. Each NVSOE node must be associated with one or both of the available network devices depending on whether path-redundancy is used.

5.7.3.1 Serial Protocols and LPC Files

Protocols determine the type of communication being done by a given network port on the board. The user assigns a protocol such as DT8 to the network port; the assigned protocol determines how messages are sent and received and determines protocol-specific application rules such as how many messages are allowed and the meaning of any “special message” bits. The CAA compiler installs software modules containing the control software for the assigned protocols into the EPROM so that the board is capable of using those protocols.

Different protocols may have different setup options. These options are defined through LPC (Link Protocol Command) files whose format depends on the protocol that uses them. CAAPE provides editing tools for creating these files; the compiler reads the files and places their contents in data structures in EPROM so that the protocol software can read the data for configuration. If the user does not specify an LPC file, the compiler uses a default file.

Refer to the appropriate protocol manuals for details on protocol operation.

5.7.3.2 IP Addresses and Network Ports

An NVSOE node which is a Client must know how to connect to a remote Server; a node which is a Server waits for each Client to request a connection, but may still need address information on the Clients in order to set up routing across subnets. A node that is neither Client nor Server must always know how to connect to its remote nodes because such nodes establish independent connections in each direction. The end point of a connection is specified by the IP address of the network device and an appropriate Ethernet socket port number value. The CAA assigns default port numbers where possible, but the user may specify the actual numeric values in cases where the protocol demands it or to override automatic assignments in special cases. In particular:

- NVSOE Clients must know the remote IP address and Ethernet port number of their Server. If the Server is on a NVSP or equivalent board, CAA input records can be used to tell the compiler that the port number is the appropriate default value; if not, the actual numeric port number must be specified.
- NVSOE Servers may have to know the remote IP addresses of the Clients in order to set up routing, but do not have to know their Ethernet port numbers.
- NVSOE nodes that are neither Client nor Server must know the IP addresses and Ethernet port numbers of their remote nodes. If the remote node is on a NVSP or equivalent board, CAA input records can be used to tell the compiler that the port number is the appropriate default value; if not, the actual numeric port number must be entered.

CAA versions 610 and later with subnets/redundancy now assign Ethernet port values from a general-usage, non-registered range of values based on protocol. Manual override is also allowed. Refer to the appropriate protocol manuals for details on protocol operation.

CAA versions before 610 assigned Ethernet port values from a different range based on the network serial port number. The Ethernet port values were 1102 plus the network serial port number. Connection to any Server, to any node neither Client nor Server, or to non-Alstom devices required manual entry of Ethernet port values.

5.7.3.3 Online Control Variable

A variable can be defined which, when True, disables message transmission from the network port.

NOTICE

The Online Control feature is not currently available for Ethernet ports. The NVSP compiler will not allow this feature at this time.

5.7.3.4 Controls and Indications

A “Control” message is one that is received by the board and an “Indication” message is a message that is transmitted by the board. Controls and indications can consist of a list of Boolean variables, or can be a buffer for sending or receiving text data.

The allowable number and size of controls and indications is determined by the protocol. In some protocols, a number of messages can be sent or received by the same network port, for example, for multidrop operation. Messages sent or received by the same port are distinguished using a binary address value of 1 to 32 bits length. Acceptable address sizes and values are determined by the protocol.

Control message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a placeholder for unused message bits.

Any Boolean variable type or array element can be used as an Indication message bit. PERMZERO and PERMONE can be used to send fixed True or False values or as placeholders for unused message bits.

Some protocols may require that control and indication messages be a multiple of some number of bits, most commonly eight. In such cases, the compiler automatically pads the message with enough PERMZERO values to reach the necessary number of bits. For example, if the user enters 14 bits and the protocol requires that all messages be a multiple of eight bits, the compiler adds two extra PERMZERO values.

5.7.3.5 Unlatched Controls

The Unlatched Controls option causes the board to automatically clear the message data received on the network port after one application cycle. This automatically sets receive variables back to False states. The application must make use of the received data within one cycle, or latch it in a separate variable for later use.

Whether or not to use Unlatched Controls is a default attribute for each type of protocol, but can be overridden by the user.

5.7.3.6 Text Messages

Certain protocols have the capability of handling text character messages. For these protocols, a method is provided for defining text messages.

5.7.3.7 Special Messages

Special messages are lists of variables used by the application to control the operation of a network port's protocol or to monitor its status. The number and meaning of variables in a special message depend on the protocol being used. Refer to the appropriate protocol manual for details.

5.7.3.8 Stations and Message Order

Some protocols may require messages to be assigned into "stations". This may be done using common addresses, or based on the order in which the messages are encountered in the input file.

5.7.3.9 MAC TCP Panel Port

This is a special type of network port that can be defined for communicating local control panel message data with MMS through one of the network devices. It uses a simplified protocol that only requires control and indication messages.

5.7.3.10 Coordination of NVSOE Input Files

Several files are used to specify data for NVSOE:

- .NSS file: NVSOE node (network serial port) declarations for the local system, including message data. Contains IP address and other network settings for CAA versionsbefore 610 without subnets/redundancy; for CAA versions 610 and later, network information has been moved to the files listed below.
- .NCW file: network links data, could be shared across entire network. Available only for CAA versions that support subnets and redundancy.
- .NNS file: IP address and port information on local and remote NVSOE nodes, could be shared across entire network. Available only for CAA versions that support subnets and redundancy.
- .GW file: IP addresses of gateways used to access remote nodes, could be shared across entire network. Available only for CAA versions that support subnets and redundancy.

Records in the .NSS file identify the NVSOE nodes (network ports) and their messages in the system being compiled. Nodes are identified by a combination of the CSEX4 board ID and the numeric value of the NVSOE node.

```
NETWORK PORT 1 = TYPE(DT8 SLAVE) , UNLATCHED CONTROLS
CONFIGURATION FILE = Q1_DT8.LPC


CONTROL = ADDRESS(00000100),LENGTH(2)
1 = 112_NZSV-ASI
    …
```

Records in the .NCW file identify the attributes of a send or receive link; SOURCE and DESTINATION records identify the senders and receivers of a message. The protocol designations for the NVSOE nodes in the local system must be consistent with those listed in the .NSS file. Note that, in the following example, the board ID “MY-NVSP” is combined with the node number to give an identifying name for the local node.

```
NVSOE LINK = TYPE(DT8 SLAVE)
LOCAL = MY-NVSP : 1, CLIENT
REMOTE = REMOTE-NVSP : 5, SERVER


NVSOE LINK = TYPE(DT8 SLAVE),REDUNDANT
LOCAL = MY- NVSP : 3, CLIENT
REMOTE = REMOTE-DEVICE, SERVER, GATEWAY(GW-1,GW-2)
```

Records in the .NNS file identify the network properties of the nodes. The nodes used by a given system are identified by matching their NAME fields to the board ID names in the .CSI file plus the node numbers in the .NSS file for local nodes, or to names in the .NCW file for remote nodes.

```
NVSOE = NAME(MY-NVSP : 1),IP(172.16.21.17),PORT(ENET1-N)
NVSOE = NAME(REMOTE-NVSP : 5),IP(172.16.21.55),PORT(ENET2-N)
```

Records in the .GW file identify the network properties of any gateways used in the network. The gateways used by a given system are identified by matching their NAME fields to the names in the source files where gateways are used.

```
GATEWAY = NAME(GW-1), IP(172.11.23.45)
```

5.8 VITAL SERIAL LINKS AND BLOCKS

⚠WARNING

VITAL COMMUNICATIONS REQUIRE UNIQUE LINK AND BLOCK SETTINGS

Failure to properly assign, maintain and control unique Link and Block settings for Vital communications within iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

The message link and block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link and Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

CRG boards, GTP boards, and VSOE nodes use sets of numerical values, called codeword sets, to ensure Vital communications. Some values encode the results of certain communication processes to ensure that they were done correctly, and others represent the actual true and false message bit parameter values. The former are called link codewords and the latter block codewords.

These Link and Block codeword sets, defined by specific numbers in the application, must be consistent on both sides of a serial link between communicating systems and devices, and must not be duplicated within a system. The CAA compiler enforces uniqueness of codeword sets, but consistency between systems is outside its scope. For this reason, the user must be responsible for assigning the codeword sets.

Link and Block numbers are defined by the user for each message and are used by the compiler to select the NISAL codewords that make up the message. Their purpose is to:

- Ensure that message codewords are unique for every board in the system. Therefore, if the wrong VSC boards or VSOE nodes are physically connected, the received message is not mistaken as the correct one leading to unpredictable results. Block numbers must not be duplicated for any message sent or received by a given iVPI system; Link numbers must not be duplicated within a given system (the compiler ensures this), and should not be duplicated within any group of systems where there is a possibility that an incorrect physical connection could be made.
- Ensure that the same codewords are used on either side of the link so that each side is able to decode the message it receives. The Link and Block numbers for a given message must be the same on each side of the link. For example, if a message output from one board has a transmit link of 3 and transmit block of 5; the remote board receiving the message must use a receive link of 3 and a receive block of 5.

5.8.1 Link Numbers

A link number from 1 to 200 represents each set of link codewords. The user is responsible for selecting a link number for each received or transmitted message. Each link number assigned must be unique for the entire iVPI system. When iVPI systems are interconnected using VSOE links, link number restrictions apply at both ends of the link. For example, if link number 1 is assigned to a VSOE message, then that link number cannot be reused in the iVPI system that sends the message nor in the iVPI system that receives it.

If CRG boards are used in an iVPI system, their messages are automatically given link numbers. These numbers are then not available for other types of messages in that system.

CRG Board Number	Links Used
1	41,42
2	39,40
3	37,38

5.8.2 Block Numbers

Codeword sets are assigned to message bits in consecutive blocks. The smallest block size corresponds to ten message bits. The number of blocks required for a message is therefore the number of blocks that covers all its message bits at ten bits per block. For example, a message with 43-bits requires five codeword blocks.

Each 10-bit codeword block is considered a sub-block in a larger "major" block: twenty consecutive sub-blocks form a 200-bit major block.

Codeword blocks are specified in "major-block.sub-block" format, where in general major-block values range from 0 to 17 and sub-block values range from 00 to 19. There are two exceptions: major-block 0 starts at sub-block 01 rather than 00, and major-block 17 ends at sub-block 00. The user specifies a starting major-block and sub-block, and the CAA assigns as many codewords as necessary to cover the entire message length. Codewords are assigned in reverse order. If a major-block boundary is crossed, the CAA starts assigning codewords from the previous one. For example, if a 43-bit message starts at block 2.01, codewords are assigned from five blocks: 2.01, 2.00, 1.19, 1.18 and 1.17.

The sub-block need not be specified. For example, if the user enters block = 2, the CAA takes the codeword block starting point to be 2.00.

The same codeword block must not be used by two messages in any given iVPI system. The CAA enforces this rule by flagging block overlaps as errors.

If CRG boards are used, certain codeword blocks are assigned automatically and are therefore not be available for other types of messages.

The list below summarizes available codeword blocks. Message bits are assigned from the user-selected starting point using as many major-blocks and sub-blocks as necessary from top to bottom toward the end of the table. Codeword blocks automatically assigned to CRG boards are indicated by *CRGn*, where “n” is the CRG board number from 1 to 3.

Major	**Sub**
17	.00
16	.19
	…
	.00
15	.19
	…
	.00

14 .19
…
.00
13 .19
…
.00
12 .19
…
.11
.10
…
.00
11 .19
…
.00
10 .19
…
.00
9 .19
…
.00
8 .19
…
.01
.00
7 .19
…
.00
6 .19
…
.00
5 .19
…
.00
4 .19
…
.00

3	.19		
		…	
		.00	
2	.19		
		…	
		.00	
1	.19		
		…	
		.07	CRG3
		…	
		.00	CRG3
0	.19		CRG3
	.18		CRG2
		…	
	.10		CRG2
	.09		CRG1
		…	
		.01	CRG1

5.9 NON-VITAL SERIAL COMMUNICATIONS

Non-Vital serial communications are the serial communications for the serial ports on NVSP boards.

5.9.1 Serial Protocols and LPC Files

Protocols determine the type of communication being done by a given serial port on the board. The user assigns a protocol such as DT8 or K2 to the serial port; the assigned protocol determines how messages are sent and received and also determines protocol-specific application rules such as how many messages are allowed and the meaning of any “special message” bits. The CAA compiler installs software modules containing the control software for the assigned protocols into the EPROM so that the board is capable of using those protocols.

Different protocols may have different setup options. These options are defined through LPC (Link Protocol Command) files whose format depends on the protocol that uses them. CAAPE provides editing tools for creating these files; the compiler reads the files and places their contents in data structures in EPROM so that the protocol software can read the data for configuration. If the user does not specify an LPC file, the compiler uses a default file.

Refer to the appropriate protocol manuals for details on protocol operation.

5.9.2 Baud Rate and Data Format

Baud rate and data format (data bits, stop bits and parity) are defined as defaults for each type of protocol, but can be overridden by the user. The user can specify specific values for baud rate and data format, and can also designate a set of Boolean variables whose values dynamically sets the baud rate as follows:

BAUD RATE CONTROL = name-3, name-2, name-1

RATE	CODE	name-1	name-2	name-3
default	000	FALSE	FALSE	FALSE
300	001	FALSE	FALSE	TRUE
600	010	FALSE	TRUE	FALSE
1200	011	FALSE	TRUE	TRUE
2400	100	TRUE	FALSE	FALSE
4800	101	TRUE	FALSE	TRUE
9600	110	TRUE	TRUE	FALSE
19200	111	TRUE	TRUE	TRUE

5.9.3 Online Control Variable

A variable can be defined which, when True, disables message transmission from the serial port.

5.9.4 Controls and Indications

A “Control” message is one that is received by the board and an “Indication” message is a message that is transmitted by the board. Controls and indications can consist of a list of Boolean variables, or can be a buffer for sending or receiving text data.

The allowable number and size of controls and indications is determined by the protocol. In some protocols, a number of messages can be sent or received by the same serial port, for example, for multidrop operation. Messages sent or received by the same port are distinguished using a binary address value of 1 to 32 bits length. Acceptable address sizes and values are determined by the protocol.

Control message variables are considered application inputs and their names cannot be declared as other variable types. PERMZERO can be used as a placeholder for unused message bits.

Any Boolean variable type or array element can be used as an Indication message bit. PERMZERO and PERMONE can be used to send fixed True or False values or as placeholders for unused message bits.

Some protocols may require that control and indication messages be a multiple of some number of bits, most commonly eight. In such cases, the compiler automatically pads the message with enough PERMZERO values to reach the necessary number of bits. For example, if the user enters 14 bits and the protocol requires that all messages be a multiple of eight bits, the compiler adds two extra PERMZERO values.

5.9.5 Unlatched Controls

The Unlatched Controls option causes the board to automatically clear the message data received on the port after one application cycle. This automatically sets receive variables back to False states. The application must make use of the received data within one cycle, or latch it in a separate variable for later use.

Whether or not to use Unlatched Controls is a default attribute for each type of protocol, but can be overridden by the user.

5.9.6 Text Messages

Certain protocols have the capability of handling text character messages. For these protocols, a method is provided for defining text messages. Text messages in different ports can be linked so that text data received on one port is automatically sent out the other.

5.9.7 Special Messages

Special messages are lists of variables used by the application to control the operation of a serial port's protocol or to monitor its status. The number and meaning of variables in a special message depend on the protocol being used. Refer to the appropriate protocol manual for details.

5.9.8 Stations and Message Order

The iVPI CAA allows multiple control, indication and special messages to be assigned to a serial port; the actual requirements for number and type of messages depends on which serial protocol has been selected for the port.

Some protocols divide the messages into multiple "stations", where each station is a group of messages that are related according to some criterion. In some cases a protocol groups messages together if they have the same address or fall within the same range of addresses; if this is not the case, iVPI groups messages based on their order in the input file: the first station is assigned the first control, the first indication and the first special message, the second station is assigned the second of each message type, etc.

If message position is significant, make sure that the number, types and order of messages in the input file meet the requirements of the protocol. The CAAPE graphics provides a Stations dialog that can be used to ensure that the messages for a given port are output in the desired order.

5.10 DATA LOGGING

Data logging takes place on NVSP boards.

5.10.1 Data Protection and Auto Dump

The user can use DATA PROTECT to specify the minimum time period in hours and minutes before a record can be erased.

AUTO DUMP can be used to specify if and how data is automatically dumped from the data logger to the MAC port. If used, auto-dump options are:

- MEMORY END: dump logger contents when data logger memory is full. This is done when any auto dump mode other than "off" is used.
- NEW DIRECTORY: older directory contents are dumped as soon as a new directory is created.
- PERIODIC: dump contents periodically at the time interval in hours.

5.10.2 System Snapshots

This option defines the interval in hours and minutes between periodic snapshots of the system information, including non-vital inputs and outputs.

5.10.3 Types of Log Data

The user can specify what types of application data and events are logged.

These general events can be logged:

- All non-vital inputs, logged periodically or when their values change
- All non-vital outputs, logged periodically or when their values change
- Current system status
- Diagnostics message
- Error messages produced by system

These serial communications events can be logged for each serial port:

- All control messages
- All indication messages logged
- Broadcast message from office or master to all field/slave locations
- Requests for information from another device
- Requests for change in operation settings from terminal/office
- Messages unique to the protocol

Application log messages can be used to specify lists of variables whose values are logged when they change. The time stamped changes can be examined through the board's diagnostics interface. See P2509 for the MMS manual, which supports the Maintenance Management System program. Multiple application log messages can be defined. Any Boolean application variable or array element can be used in an application log message. PERMZERO and PERMONE values can be used as placeholders for unused message bits.

5.10.4 Log Data Output Options

The Print Mode option specifies how the events are to be retrieved, either First In First Out (FIFO), or Last In First Out (LIFO).

The Data Logging Interface specifies the log data is output to a VT100.

5.10.5 Data Logger Names

The user can specify that the names of application log message variables are to be stored in the board's EPROM. This uses extra memory but allows the data logger reports viewed through diagnostics to have readable names for the logged variables.

5.11 VITAL LOGIC

Vital logic is the Vital application logic used in VSP boards.

NOTICE

Statements/Equations can occupy more than one line, as long as the last symbol on each intermediate line is an operator but not a parenthesis.

5.11.1 Internal Variables

Internal variables are for internal storage and are not hardware inputs or incoming message bits. They can be used in the logic and output as hardware output or outgoing message bits. Vital logic makes use of four kinds of internal variables: Current Results, Self-Latched Parameters, Timer Parameters and Timer Programmable Parameters.

5.11.1.1 Current Results

Current result parameters are used as intermediate results of expressions, and their values are cleared at the end of each Boolean expression evaluation cycle.

5.11.1.2 Self-Latched Parameters

Self-latched parameter values are saved from a previous cycle, and can be used in expressions of this form: A = B * C + D * E * A, where the result A is also a parameter in a product term in the expression (for example, stick functions).

Self-latched parameters can also be used in a sequence of equations such as:

X = Y * Z

Y = A * B * C

where Y is a self-latched parameter. In this case, due to expression ordering a parameter value must be known before it is assigned a value so the parameter value is taken from the previous cycle.

5.11.1.3 Timer and Timer Programmable Parameters

Timer/Timer Programmable parameters are the result parameters in those expressions that evaluate to true only after some fixed time delay. Vital time delays are referred to as permanently programmed Vital timers (PPVT) or software timers. Vital time programmable delays are referred to as Field-Settable Software Vital timers (FSSVT) or software timers.

5.11.2 APPLICATION Statements

APPLICATION statements give an application name for the Boolean equation records that follow them. Information contained on this statement creates an index by page number at the end of the compiler listing. APPLICATION statements are also used by some of CAAPE's graphical logic printing features to produce an index.

5.11.3 LOCATION Statements

This statement gives a location identification for the Boolean equation records that follow it. The information contained on this statement creates an index by page number at the end of the compiler listing.

NOTICE

If APPLICATION and LOCATION statements occur in pairs (if there are no other statements between them), then the information on both statements appears on the same line of the index report. The page number used is the page number of the first statement.

5.11.4 Boolean Equation Statements

Boolean Equation data records contain the Boolean equations to be evaluated by the processor. They are evaluated in the order in which they appear in the logic.

5.11.4.1 Logic Expression

The Boolean equation solves a Boolean logic expression and places the result in one or more result variables. The logic expression is composed of one or more parameters combined by logical “and” and “or” operators. The parameters in Vital logic must be defined as inputs, outputs, current result parameters, self-latched parameters, NVSP-TO-VSP message parameters, filament check values for lamp drive outputs, or absence of current detector (CK) values for Vital outputs. Lamp drive output 'state' parameters can also be used in Vital Boolean equations.

The complement of a parameter can be used to indicate a logical “not." A lamp drive output “state parameter” cannot be complemented.

Boolean equation logic is converted to and stored in *sum-of-products* form for evaluation by the processor. Sum-of-product form consists only of sequences of and’ed parameters, with each sequence or’ed together. In relay terms, an equation in sum-of-products form consists only of a single set of branches with no sub-branches.

For example, if the logic of the equation is entered in complex form by the user as

```
BOOL X = (( A + B) * C)
```

the statement is converted and stored in the form

```
BOOL X = (A * C + B * C)
```

where ‘*’ is an AND operation and ‘+’ is an OR operation.

Each set of and’ed variables is called a *product term*. In the previous example, A * C and B * C are product terms. Vital application logic is capable of processing up to 63 product terms of 63 parameters each in a single equation.

NOTICE

No more than three Vital output state parameters can appear in a single product term.

5.11.4.2 Result List

Each result name in a Vital equation must have been previously defined as a current result, a self-latched parameter, a timer result, or a Vital output. Up to seven results per expression can be used with these restrictions on Vital Boolean equations:

1. If there is a timer expression result type, it must be the only result.
2. If there is a self-latched result, it can be paired with a maximum combined total of five output ports and current result parameters as results for the same expression. Only one self-latched result is allowed per expression.
3. There can be a maximum combined total of seven output ports and current results as results of an expression.

5.11.4.3 Boolean Equations as Relay Circuits

Boolean equations can be pictured as relay circuits, with the relay contacts (the logic expression parameters) wired to drive a relay coil (the equation result). “AND” operations are equivalent to serial connections of contacts, “OR” operations are equivalent to parallel connections, and “NOT” operations are equivalent to using a relay’s back contact.

5.11.4.4 SLOW Equations

This is a slow pick/slow drop relay equation. Three equations are generated in place of the original data.

For example, the slow pick/slow drop equation

SLOW 1L = (1ES * 1WS * 1TP)

results in the following equations being generated:

1L = (@EXPR0019-SL + @EXPR001A-SL)

@EXPR0019-SL = (@EXPR001A-SL)

@EXPR001A-SL = (1ES * 1WS * 1TP)

5.11.4.5 Fixed Interval Timer Equations

Fixed Interval Timer Equations are also referred to as Permanently Programmed Vital Timers (PPVT). In this type of equation, the result is set True a user-specified period of time after the expression evaluates to true.

NOTICE

The maximum time delay period is 59 minutes and 59 seconds.

The compiler converts each of these equations into three special-purpose equations called the initial, intermediate, and final timer equations. They use internal values to perform their functions and cannot be represented in the form of normal sum-of-products equations. The initial equation becomes true based on the user-entered equation. This starts the timer running and keeps it running as long as the equation is true. (If this equation at any time evaluates to false, the timer reverts to its full time delay.) The intermediate timer equation proves that the timer is started from the correct initial state, and the final timer equation becomes true after the timer has finished running.

For example, this equation:

```
TIME DELAY = 5 SECONDS
BOOL 1TTE = (1T-TE)
```

is converted to:

```
@PPVT0010-CR = ( 1T-TE )
@PPVT0010-LA = ( @PPVT0010-CR * @PPVT0010-LA + @PPVT0010-CR )
1TTE = ( @PPVT0010-CR * @PPVT0010-LA + @PPVT0010-CR * @PPVT0010-LA *
1TTE )
```

The equation must have a Timer Parameter as a result. In addition, a vitally delayed equation should (in general) have the Vital Relay Driver front contact, called VRDFRNT-DI, as an input parameter in every product term in the equation; this is true for every Vital timer which runs time in the event of a system restart.

5.11.4.6 Field-Settable Software Vital Timer Equations

NOTICE

These Timer types are available in iVPI CAA 611a and later.

Field-Settable Software Vital Timer Equations are also referred to as Field-Settable Software Vital Timers (FSSVT). These FSSVT Timers are very similar to the above-described Fixed Interval Timers (PPVT) in that they behave in the same way and the equations generated are the same. The difference between the two timers is that the FSSVT Timers can be changed in the field using the AlsDload utility (refer to manual AlsDload User Guide P2512B) without using CAAPE and CAA to rebuild the entire application.

In this type of equation, the result is set TRUE a user-specified period of time after the expression evaluates to true. The compiler converts each of these equations into three special-purpose equations called the initial, intermediate, and final timer equations. They use internal values to perform their functions and cannot be represented in the form of normal sum-of-products equations. The initial equation becomes true based on the user-entered equation. This starts the timer running and keeps it running as long as the equation is true. (If this equation at any time evaluates to false, the timer reverts to its full time delay.) The intermediate timer equation proves that the timer is started from the correct initial state, and the final timer equation becomes true after the timer has finished running.

For example, this equation:

```
TIME DELAY PROGRAMMABLE = 5 SECONDS
BOOL 1TTE = (1T-TE)
```

is converted to:

```
@FSSVT0010-CR = ( 1T-TE )
@FSSVT0010-LA = ( @FSSVT0010-CR * @FSSVT0010-LA + @FSSVT0010-CR )
1TTE = ( @FSSVT0010-CR * @FSSVT0010-LA + @FSSVT0010-CR * @FSSVT0010-
LA * 1TTE )
```

The equation must have a Timer Parameter as a result. In addition, a vitally delayed equation should (in general) have the Vital Relay Driver front contact, called VRDFRNT-DI, as an input parameter in every product term in the equation; this is true for every Vital timer which runs time in the event of a system restart.

The time value entered in the equation becomes the maximum value that can be programmed into the timer equation by the AlsDload utility.

5.11.4.7 Using Library Files

Library files can be used to insert generic Vital statements into the logic and customize them for an application. The process consists of identifying the library file and using LIBR statements to insert one or more of its library members into the logic and in the process substitute application-specific names for the library members' generic names.

See the CAAPE User's Guide for information on creating and editing library files.

5.11.4.8 Identifying Library Files

To identify library files, a LIBRARY PATH can be defined and/or use LIBRARY FILE statements. LIBRARY PATH can be used to specify the common path for multiple library files or to give the full path of a single library file to be used in the application. LIBRARY FILE provides a file name to be appended to the LIBRARY PATH if it exists, or can itself give the full path to the library file. In addition, the LIBR statement which names the library member can include a library file name. In all cases, if a file extension is not specified an extension of '.LIB' is assumed.

NOTICE

The ADV does not access the .VPC file, so use of the LIBRARY PATH record for Vital applications is not recommended. The ADV is unable to find the library file.

Example 1: a single library file, LIB1.LIB, with two members.

```
LIBRARY PATH = E:\LIBFILES\LIB1
…
LIBR MEMBER1
LIBR MEMBER2
```

or

```
LIBRARYFILE = E:\LIBFILES\LIB1
LIBR MEMBER1
LIBR MEMBER2
```

Example 2: two library files, LIB1.LIB and LIB2.LIB, with one member each.

```
LIBRARY PATH = E:\LIBFILES
…
LIBRARY FILE = LIB1
LIBR MEMBER1
LIBRARY FILE = LIB2
LIBR MEMBERA
```

or

```
LIBRARY FILE = E:\LIBFILES\LIB1
LIBR MEMBER1
LIBRARY FILE = E:\LIBFILES\LIB2
LIBR MEMBERA
```

or

```
LIBRARY PATH = E:\LIBFILES
…
LIBR LIB1 : MEMBER1
LIBR LIB2 : MEMBERA
```

5.11.4.9 Inserting Library Members

Use a LIBR statement to insert the contents of a library member into the logic and substitute application names for its generic names.

For example, if library member TEST_LOGIC has generic name DIR and consists of logic statement:

```
BOOL [DIR]-LOCKED = ( [DIR]-READY * X )
```

then using:

```
LIBR TEST_LOGIC( EAST )
```

substitutes EAST for generic name DIR and replaces the LIBR statement with:

```
BOOL EAST-LOCKED = ( EAST-READY * X )
```

5.12 NON-VITAL LOGIC

Non-Vital logic is the non-vital application logic used in NVSP boards.

NOTICE

Statements/Equations can occupy more than one line, as long as the last symbol on each intermediate line is an operator but not a parenthesis.

Also, when writing IF and WHILE logic, it is best not to put two statements on the same line because REDP and the Simulator may not interrupt it correctly.

For example, instead of formatting IF and WHILE statements as:

```
IF (….) X = A + B
```

Write as:

```
IF (….)
 X + A + B
```

Or

```
IF (….)
{ X + A + B }
```

5.12.1 Constant Declarations

The user can define readable text names that represent numerical values used in array declarations or logic statements. When the constant name is encountered in the logic, its associated number is substituted.

For example, if a constant ARRAY_SIZE is defined as 10, when the array variable declaration:

```
ARRAY_VAR[ARRAY_SIZE]
```

is encountered the compiler defines ARRAY_VAR to be of size 10.

5.12.2 Internal Variables

There are three types of internal variables: Boolean Parameters, Integer Parameters and Timer Parameters.

5.12.2.1 Boolean Parameters

Boolean parameters are internal True or False values in the NVSP program (not inputs or outputs), and they are also referred to as internal parameters. All Boolean parameter names must be unique. Arrays of Boolean parameters can be declared; array size can be a numeric value or a constant.

For example:

```
BOOLARRAY[5]
BOOLARRAY2[ ARRAY_SIZE ]
```

5.12.2.2 Integer Parameters

Integer parameters are internal to the NVSP program (not inputs or outputs), and contain numeric rather than True or False values. All Integer parameter names must be unique. Arrays of Integer parameters can be declared; array size can be a numeric value or a constant.

5.12.2.3 Timer Parameters

Timer parameters are internal to the NVSP program (not inputs or outputs), and are the results of time delay equations. All Timer parameter names must be unique.

5.12.3 Non-Vital Logic Features

Non-Vital logic includes a number of advanced programming features that supplement simple Boolean logic. These include:

- Integer arithmetic
- Boolean logic (TIME DELAY, BOOL, APPLICATION)
- Program flow control (IF..ELSE, WHILE, GOTO, CALL)
- Predefined subroutines
- Predefined variables

5.12.4 Arrays

Boolean and integer arrays can be used in non-vital logic. Boolean array elements can also be used in serial output messages. Array size and index information can be expressed either directly as a number, or as a previously-defined constant. The first element of an array has an index of zero.

Arrays can be indexed by a fixed numerical value, an integer variable name, or an integer expression.

For example:

X[1]

X[VAR1]

X[INTVAL + 2]

Individual array elements or entire arrays can be passed to subroutines.

5.12.5 Application Logic Execution Errors

Possible errors that can occur while application logic is being executed include incorrect array indexing and attempts to divide a number by zero. The compiler automatically generates code to check for these conditions and, if found, update two predefined variables:

- Boolean variable APP_ERROR_FLAG is set to TRUE.
- Integer variable APP_ERROR_STMT is loaded with the number of the statement where the error occurred.

This code is not generated for fixed-index array addressing, since array bounds can be checked at compile time in this case. The user can disable generation of error checking code by using the NOERRCODE run control. This might be done if the user was confident that the logic generates no application errors at run time, and wanted to save the PROM memory required for error checking (about 20 bytes per array reference, 10 bytes per divide operation).

5.12.6 Integer Arithmetic

Simple four-function arithmetic can be performed using signed 16-bit integer variables and constants. Values can range from -32767 to 32767.

5.12.6.1 Integer Variables

Integer variable names must be declared before being used, or be named as arguments in the definition of a subroutine. Each name can be up to 16 characters in length, including letters, numbers, dashes and underscores; the first character in the name must be a letter or number.

For example:

10X, TRAIN-COUNT, RESULT_1

5.12.6.2 Integer Constants

Integer constants can be entered in decimal or hexadecimal format. Decimal entries can range from -32767 to 32767. Hexadecimal entries are indicated by preceding the number with 0x and can range from 0x0000 to 0xFFFF.

For example:

32, -128, 0x12F

5.12.6.3 Integer Operators

The four standard arithmetic operators +, -, * and /, as well as the unary operator "-" (negate) are allowed. Variable names are allowed to have embedded dashes, so the subtract operator must be separated from any name by at least one blank space.

If an overflow or underflow occurs, the variable is truncated at 16 bits; for example, 0XFFFF + 1 = 0X0000. Attempts to divide by zero leaves the dividend unchanged; for example, 10 / 0 = 10. No error messages are produced.

Operator precedence is:

Symbol	Operation	Precedence
-	negate	1
*	multiply	2
/	divide	2
+	add	3
-	subtract	3

Example use of "-" in integer expressions:

```
MATH-RESULT = MATH-PARAM-1 - MATH-PARAM-2
```

5.12.6.4 Integer Equation Statements

These statements are the integer equivalent of Boolean equations. An arithmetic expression involving one or more integer variables and/or constants is evaluated and the result stored in up to seven integer result variables. All variables must be declared prior to use. An integer equation can occupy more than one line, as long as the last symbol on each intermediate line is an operator.

Integer array elements can be used in expressions and as results.

For example:

```
A,E =  VAR1 * (VAR2 + VAR3) +
            VAR4 * (VAR5 + VAR6)
X,Y,Z = 65
M[20] = VALS[X] + 5
```

5.12.7 Boolean Logic

5.12.7.1 Boolean Variables

This category includes all valid inputs and outputs as well as variables declared in the Boolean Parameter Section described earlier and Boolean arguments in subroutine definitions.

5.12.7.2 Boolean Constants

Either TRUE or PERMONE can be used for True conditions; either FALSE or PERMZERO can be used for False.

For example:

```
BOOL FLAG1 = PERMZERO
BOOL BOOLVAL = TRUE
```

5.12.7.3 Boolean Operators

The same operators used in Vital logic are used here. The meanings of '*' and '+' can be AND/OR or MULTIPLY/DIVIDE depending on whether they appear in a Boolean or an integer equation.

Operator precedence is as follows:

Symbol	Operation	Precedence
.N.	NO	1
*	AND	2
+	OR	3

5.12.7.4 Boolean Equation Statements

These are the standard BOOL statements previously described for Vital logic. An expression involving one or more Boolean variables and/or constants is evaluated and the result stored in up to seven Boolean result variables. Boolean array elements can be used in expressions and as results.

Unlike in Vital logic, the logic expression is not converted to sum-of-products form for storage in EPROM. Instead, it is stored in a format that is meant to maximize speed of execution.

For example:

```
BOOL RES1,RES2 = (A * .N.B +
            C * (D + E))
BOOL PARM1 = FALSE
BOOL X[2] = Y[4] * Z
```

5.12.7.5 TIME DELAY Statements

WARNING

PROTECT VITAL TIMER EQUATIONS WITH VRDFRNT-DI

Vital Boolean and timer equations are evaluated in every one-second application cycle regardless of the state of the VRD. Therefore every timer equation must include the VRDFRNT-DI Vital input as a constituent in order to prevent the timer from running short and completing an evaluation of the equations prematurely.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

If a Boolean equation is preceded by a TIME DELAY specification, then code is generated for a time delay of the specified amount based on the expression value. A time delay equation starts False, waits for the specified time delay amount when the logic expression becomes True, and goes True only if the expression remains True for the specified time.

Time delay specifications can be made in HOURS, MINUTES, SECONDS and MSEC. It is also possible to name an integer variable in the TIME DELAY; in this case, the contents of the variable are used to set the time delay amount when the timer starts timing. In either case, the maximum time delay period is 6535 seconds (about 108 minutes) in 100-millisecond increments.

For example:

```
TIME DELAY = 1 HOUR, 10 SECONDS
BOOL TMR = (X * Y)

TIME DELAY = TIMEOUT_VAL
BOOL TMR-2 = (A + B +C)
```

5.12.7.6 APPLICATION Statements

These are for documentation purposes only and are ignored by the program.

For example:

```
APPLICATION = ROUTE LOCKING
```

5.12.8 Program Flow Control

The sequence of non-vital logic statements executed can be controlled by conditional logic (IF..ELSE and WHILE), unconditional jumps (GOTO), and subroutine calls (SUBROUTINE, RETURN, CALL).

5.12.8.1 Conditional Logic

IF..ELSE and WHILE statements conditionally execute sections of non-vital logic.

5.12.8.2 Logical Expressions

Logical expressions must evaluate to a True or False result. Logical expressions can include arithmetic operations, so the symbols for logical AND and OR are different from those in Boolean equations.

Precedence in logical expressions is as follows:

Symbol	Operands	Precedence	Operation
!	1 Logical	1	complement (NOT)
-	1 Integer	1	complement (negate)
*	2 Integer	2	multiply
/	2 Integer	2	divide
+	2 Integer	3	add
-	2 Integer	3	subtract
>	2 Integer	4	greater than
<	2 Integer	4	less than
>=	2 Integer	4	greater than or equal to
<=	2 Integer	4	less than or equal to
==	2 Logic/Int	5	equals
!=	2 Logic/Int	5	not equal
&&	2 Logic	6	logical AND
||	2 Logic	7	logical OR

For example:

```
A && (B || C)
!A && (INT1 + 10 <= INT2)
```

5.12.8.3 IF..ELSE Statements

The IF..ELSE statement allows conditional transfer of control dependent on a logical result.

The statement format is:

```
IF( logical-expression )
        statements-list
ELSE
        statements-list
```

- *statements-list* can be a single statement or multiple statements enclosed in { } brackets.
- The ELSE clause is optional.
- IF..ELSE statements can be nested up to 16 deep.
- If *logical*-expression evaluates to True, the statement list following the IF is executed.
- If *logical*-expression evaluates to False, the ELSE statement list is executed if it exists; if there is no ELSE, execution jumps to the first statement following the IF statement list.

For example:

```
IF (A==B)
                BOOL C = A*Y+Z

IF (TEST1 && !TEST2)
{
                COUNT = COUNT + 1
                IF (COUNT==10)
                                TEST1 = FALSE
}
ELSE
{
                COUNT = 0
}

IF (VAR1)
                BOOL X = X*Y
ELSE IF (VAR2)
                BOOL Y = Y*Z
```

5.12.8.4 WHILE Statements

These statements cause a block of non-vital logic to be executed repeatedly as long as a conditional logic expression evaluates to TRUE.

The statement format is:

```
WHILE ( cond-expression )
        statement-list
```

- *cond-expression* is the expression to be evaluated.
- *statement-list* is the block of logic that executes as long as *cond-expression* is True.
- *statement-list* can be a single statement or multiple statements enclosed in { } brackets.

For example:

```
WHILE( I<10 )
{
    BOOL Y[I] = FALSE
    I = I + 1
}
```

5.12.8.5 Statement Labels

Labels identify a location in the logic for documentation or for transfer of control by a GOTO statement. Labels are up to 16 characters long, must be unique, and are followed by a colon:

For example:

```
LABEL:
```

5.12.8.6 GOTO Statements

The GOTO statement allows transfer of control to a point in the program identified by the accompanying label name. Jumps cannot be made between subroutines, or between a subroutine and the main program.

For example:

```
LABELNAME:
N = N+1
IF(N<10)
        GOTO LABELNAME
```

5.12.8.7 SUBROUTINE Definitions

All subroutines must be defined prior to the main body of non-vital logic; definitions cannot be nested.

The subroutine format is:

SUBROUTINE subroutine-name (argument-definition-list)

statement-list

RETURN

END subroutine-name

- *argument definition list* is optional and consists of one or more argument definitions separated by commas.
- *subroutine* argument is a local variable which is passed some value when the subroutine is called and can be used only in equations within that subroutine: its "scope" is local to the subroutine. Variables declared prior to the non-vital logic section have a scope of MAIN and can be used anywhere. A subroutine argument is defined by giving its type followed by its name. Allowed types are BOOL, INT, *BOOL, *INT. Subroutine arguments cannot have the same name as variables defined prior to the non-vital logic, but can have the same name as arguments of another subroutine. The argument list can occupy multiple lines as long as each line ends with a comma.
- BOOL and INT arguments are passed the ADDRESS of the corresponding Boolean or integer variable named in the subroutine CALL. If their values are changed within the subroutine, then the values of their corresponding CALL variables change also. To identify an entire array passed to the subroutine, add empty '[]' brackets after the argument name.
- *BOOL and *INT arguments are passed the VALUE of corresponding Boolean or integer variables, constants or expressions in the subroutine CALL. Changes to these variables do not extend outside the subroutine. They can be used to merely inform the subroutine of a variable's value without changing it.

Multiple RETURN statements are allowed within a subroutine; one subroutine END is required. An explicit RETURN statement is not required just before the END; the CAA automatically inserts one.

See CALL Statement for examples.

5.12.8.8 CALL Statements

This statement allows transfer of control to a subroutine.

The statement format is:

```
CALL subroutine-name ( argument-list )
```

- *argument list* is optional and can contain one or more Boolean or integer variables or constants and/or logical or integer expressions separated by commas:
  - the number and type of arguments must match those in the definition of the subroutine.
  - since arithmetic operations are allowed in the argument list, logical expressions cannot use the '*' and '+' symbols used in Boolean equations; they use '&&' and '||' instead.

Example 1: subroutine with no arguments:

```
SUBROUTINE NO-ARGS-SUB
        BOOL GLOBAL-NAME = TRUE
END NO-ARGS-SUB
…
CALL NO-ARGS-SUB
```

Example 2: argument list and parameter passing:

```
SUBROUTINE COUNTSUB(BOOL TIMEOUT-FLAG,
                                        INT COUNTER,
                                        *INT INCREMENT,
                                        *BOOL UPCOUNT-FLAG)
        IF (UPCOUNT-FLAG==TRUE)
                        COUNTER = COUNTER + INCREMENT
    ELSE
                        COUNTER = COUNTER - INCREMENT
        IF (COUNTER==0)
                        BOOL TIMEOUT-FLAG = TRUE
    ELSE
                        BOOL TIMEOUT-FLAG = FALSE
END COUNTSUB
…
CALL COUNTSUB(TIMED-OUT, COUNT, 5, TRUE)
```

Example 3: scope of changes to argument variables, use of return:

```
SUBROUTINE TESTER( INT ADDRESS-PARM, *INTVALUE-PARM)
        IF( ADDRESS-PARM == 0XFFFF )
                                    RETURN
    ELSE
    {
                                    VALUE-PARM = VALUE-PARM + 1
                                    ADDRESS-PARM = VALUE-PARM
    }
END TESTER
…
INT1 = 0
INT2 = 0
INT3 = 0
CALL TESTER(INT1,INT2)
…
CALL TESTER(INT3,20)
(INT1 now contains 1, INT2 still contains 0, and INT3 contains 21.)
```

Example 4: same name for arguments in more than one subroutine, expression passing:

```
SUBROUTINE X( BOOL A )
        BOOL A = TRUE
END X
SUBROUTINE Y( *BOOL A )
        BOOL FLAGQ = A
END Y
…
CALL X(FLAG1)
CALL Y( !FLAG1 && (FLAG2 || FLAG3))
```

Example 5: entire array passed to a subroutine

```
SUBROUTINE SUBRT( BOOL X[] )
        BOOL X[3] = FALSE
        BOOL X[5] = TRUE
END SUBRT
…
BOOL BOOLARRAY[3] = TRUE
CALL SUBRT( BOOLARRAY )
```

5.12.9 Predefined Subroutines

Predefined subroutines can be called in the user's application logic, but need not be defined; the CAA automatically generates code for their functions.

5.12.9.1 Predefined Timer Subroutines

Predefined timer subroutines are built-in functions which start, stop, and check the status of the timer associated with any timer variable previously declared in the Timer Parameter Section. They provide extra timing flexibility, including timing of other functions beside Boolean equations.

The subroutine format is:

TIMER_START (BOOL timer_var, *INT delay_size)—This function starts the timer associated with timer_var, first initializing it to delay_size. Delay_size is in 100 millisecond increments.

TIMER_STOP (BOOL timer_var)—This function stops the timer associated with timer_var.

TIMER_STATUS (BOOL timer_var)—This function sets timer_var to FALSE if its timer is not started or is started and still running; if not, timer_var is set TRUE.

For example:

```
IF( START-PULSE ){
    BOOL OUTPUT-BIT = TRUE
    BOOL START-PULSE = FALSE
    BOOL PULSE-IN-PROG = TRUE
    CALL TIMER_START(TIMER1,20)
}
IF( PULSE-IN-PROG ){
    CALL TIMER_STATUS(TIMER1)
    IF ( TIMER1==TRUE ){
    BOOL OUTPUT-BIT = FALSE
    BOOL PULSE-IN-PROG = FALSE
    }
}
IF( ABORT-PULSE ){
    CALL TIMER_STOP(TIMER1)
    BOOL OUTPUT-ON = FALSE
    BOOL PULSE-IN-PROG = FALSE
    }
```

5.12.9.2 Predefined Data Input/Output Subroutines

The predefined data input/output subroutines function format is:

GET_INPUTS (address of Boolean inputs_received flag) - This function acknowledges new data from the NVSP system routines, allowing them to load more data when received.

UPD_OUTPUTS() - This function transfers new data to the NVSP system routines for physical output.

The CAA automatically generates a main program loop starting with a call to GET_INPUTS and ending with a call to UPD_OUTPUTS. These routines MUST be executed periodically for new data to be processed: the programmer should insert them whenever bypassing these points using GOTOs; especially when waiting for new data inputs.

For example:

User enters:

```
SUBROUTINE FIRSTSUB
    …
END FIRSTSUB
SUBROUTINE LASTSUB
    …
END LASTSUB
BOOL X = Y+Z
```

CAA generates:

```
SUBROUTINE FIRSTSUB
    …
END FIRSTSUB
SUBROUTINE LASTSUB
    …
END LASTSUB
LABEL001: CALL GET_INPUTS(@INDATA-BP)
BOOL X = Y+Z
CALL UPD_OUTPUTS
GOTO LABEL001
```

GET_DIP_SWITCH (integer parameter name) - This function retrieves the NVSP DIP switch values and returns the value as a 16 bit integer value. The switch value is returned as:

integer bit:15 .....8 7.....0

switch: APP16 ... APP9 APP8 .. APP1

For example:

```
**************************************************
    INTEGER PARAMETER SECTION
**************************************************
DSWITCH


**************************************************
BOOLEAN EQUATION SECTION
**************************************************
CALL GET_DIP_SWITCH( DSWITCH )
```

5.12.9.3 Predefined Data Conversion Subroutines

Predefined data conversion subroutines can be used to change the form of certain types of data.

The subroutine format is:

CALL BIN_TO_INT(INT intaddr, *BOOL boolval_1,...,*BOOL boolval_n) - This subroutine converts a list of up to sixteen Boolean values into an integer, storing the result in variable 'intaddr'. The least significant Boolean value should be leftmost in the argument list.

For example, this code results in the binary value 100 being stored in INTNUM:

```
BOOL BIT0 = FALSE
BOOL BIT1 = FALSE
BOOL BIT2 = TRUE
CALL BIN_TO_INT(INTNUM,BIT0,BIT1,BIT2)
```

CALL BCD_TO_DEC(INT decaddr, *INT bcdval) - This subroutine converts a BCD value 'bcdval' to decimal and stores the result in 'decaddr'.

For example, this code results in the decimal value 1234 being stored in INTNUM:

```
BCDNUM = 0x1234
CALL BCD_TO_DEC( INTNUM,BCDNUM )
```

CALL DEC_TO_BCD(INT bcdaddr, *INT intval) - This subroutine converts a decimal value 'intval' to BCD and stores the result in 'bcdaddr'.

For example, this code results in the BCD value '99' being stored in BCDNUM:

```
CALL DEC_TO_BCD( BCDNUM,99 )
```

CALL INT_TO_BIN(*INT intval, BOOL booladdr_1,...,BOOL booladdr_n) - This subroutine converts a decimal value 'intval' into a string of bits, storing them in a list of up to sixteen Boolean variables. The least significant bit is stored in the leftmost Boolean variable.

For example, this code sets BIT0 false, BIT1 false, and BIT2 true:

```
INTNUM = 4
CALL INT_TO_BIN( INTNUM,BIT0,BIT1,BIT2 )
```

5.12.9.4 Predefined Date/Time Subroutines

On systems equipped with a real-time clock, these subroutines can be used to access the current date and time.

CALL GET_DATE(INT day, INT month, INT year) - This subroutine returns current date used by the hardware's real-time clock (RTC). The RTC provides data values in Binary Coded Decimal (BCD) format. Day values range from 0x0001 to 0x0031. Month values range from 0x0001 to0x0012. Year values range from 0x0001 to 0x9999. For example, a hexadecimal value of 0x0012 represents the month of December.

```
CALL GET_DATE ( CX1-HOUR, CX1-MONTH, CX1-YEAR )
```

CALL GET_TIME(INT second, INT minute, INT hour) - This subroutine returns current time used by the hardware's real-time clock (RTC). The RTC provides time values in Binary Coded Decimal (BCD) format. Second values range from 0x0000 to 0x0059. Minute values range from 0x0000 to 0x0059. Hour values range from 0x0000 to 0x0023.

```
CALL GET_TIME ( CX1-SEC, CX1-MIN, CX1-HOUR )
```

5.12.10 Predefined Variables

Predefined variables can be used to:

- Inform the non-vital application logic when data has been read from or written to the VSP board.
- Control latching of momentary values in NVSP-to-VSP message data.

It is NOT necessary to declare these variables before they are used in the logic: they are CAA-defined variables.

Although these variables are controlled by the process of transferring data between NVSP and the VSP board, it is NOT guaranteed that their values are changed in consistent one second intervals. They should not be used to check for or count one second periods.

5.12.10.1 DPRAM_READ

When DPRAM data is available from the VSP, the NVSP system software places it in a RAM buffer and sets an associated flag to 'Data Available'. At the start of its cycle (before the first equation is processed) the application detects that data is available and changes the buffer flag to 'Data Accepted'. This flag change is for the benefit of the system software only; the process of placing the data in the buffer makes it immediately available to the application logic.

The DPRAM_READ variable is set when the application detects that data is available in the buffer. The variable is kept true for one non-vital application cycle (not the one-second iVPI cycle, but the time between successive calls to the non-vital application routine that handles input data from the system) and then cleared.

This variable can be used to detect when data has just been received from the VSP.

5.12.10.2 DPRAM_WRITE

When transferring data from NVSP to the VSP board, the application writes data to a RAM buffer and sets an associated flag to 'Data Available'. When it is time to write to the VSP, the NVSP system software sets the buffer flag to 'Data Accepted' and transfers the buffer data to DPRAM.

DPRAM_WRITE is set true when NVSP-to-VSP buffer data has been accepted by the system software. The application keeps the variable true for one non-vital application cycle (again, not the one-second VSP cycle but the time between successive calls to the non-vital application routine that outputs data to the system) and then clears the variable.

Two additional changes were made to the application:

- If NVSP-to-VSP data changes it is loaded immediately to the buffer whether the old buffer data has been taken. This can be overridden by setting a latch enabling variable true. See next section for details on the latch variable.
- When old data has been accepted, the buffer flag is set to 'Data Available' whether or not NVSP-to-VSP data has actually changed. This ensures that DPRAM_WRITE goes true once every one-second VSP cycle.

This variable can be used to detect when data has just been sent to the VSP board. It is useful for manually latching momentary NVSP-to-VSP values until they have been sent, and for detecting when NVSP-to-VSP communications has failed (DPRAM_WRITE fails to toggle once a second). Application logic may sometime need to check for two successive DPRAM_WRITE transitions before unlatching its NVSP-to-VSP data. If the logic set the new data just when the previous data had been taken, DPRAM_WRITE would be true for that cycle and would then go false before the new data was actually taken. The application logic would have to wait for the next time DPRAM_WRITE went true before unlatching the data.

5.12.10.3 DPRAM_LATCH

This flag can be used to enable latching of true NVSP-to-VSP values until they have been accepted by the system and sent to the VSP. This prevents any momentary true value from being lost, and requires less logic than latching individual bits with DPRAM_WRITE.

To latch true NVSP-to-VSP values until they have been sent to the VSP, set the DPRAM_LATCH variable true. Be aware that if latching is enabled, momentary false values may be lost. These can still be latched individually using DPRAM_WRITE if desired.

If it is determined that NVSP-to-VSP communications has failed, DPRAM_LATCH can be cleared to flush any true values from the buffer.

5.12.11 Using Library Files

See SECTION 5.11.4.7 – Using Library Files; library file usage is the same in both Vital and non-vital logic.

SECTION 6 – INPUT FILE ORGANIZATION

This section discusses general rules and recommendations for organizing input files.

6.1 INPUT DATA SECTIONS

Input data is organized into sections that describe different areas of application functionality. For example, the VSP Communications Section describes data passed through DPRAM between the Vital and the non-vital applications in an iVPI system; the Boolean Equation Section describes the application logic.

6.2 MODULE SECTIONS

If a hardware module uses a split motherboard, each iVPI system within the module is handled separately. Separate input files and separate compiles are required for each system.

6.3 MAIN / INCLUDE FILES

When a compiler is started it is passed the name of the main input file for the application. This file must have an extension of .VPC for Vital applications, .VCC for Vital Comm applications, and .CSI for non-vital applications.

The main input file includes documentation records, and generally contains references to other files which contain the rest of the data. File references are in the form of INCLUDE records format:

INCLUDE *filename*

- *filename* is the name of the file to be included in the compile.

When the compiler encounters an INCLUDE record, it opens the include file and starts reading it. When the compiler has finished reading the included file, it closes that file and resumes reading the main file on the next line after the INCLUDE record.

Only one level of INCLUDE files is allowed; included files cannot in turn contain other INCLUDE records. Aside from this limit, the compiler is indifferent as to whether or how these INCLUDE records are used. All the input records could be put on the main file with no INCLUDE records at all, or a section of input data could be split between the main file and multiple included files. However, this manual recommends a specific INCLUDE file organization based on:

- The need to share certain data such as hardware definitions and VSP Communications between the Vital and non-vital applications which use it. Sharing input files allows the user to write the data once rather than having to copy it for each application, and simplifies its maintenance when changes are made.
- Devoting a separate file to each major section of input data, avoiding the need to edit a single large file and thus making the data easier to handle.

iVPI CAA versions from the 600 series allow INCLUDE file names with embedded spaces; such names must be enclosed in quotes, for example: INCLUDE “MY IVPI APP.HDW”.

6.4 SYSTEM COMPILER INPUT FILES

The recommended INCLUDE file organization is shown in Table 6–1.

Table 6–1. Recommended INCLUDE File Organization for iVPI System Compiler

File Type	Description
.HDW	Hardware file, shared by Vital and non-vital applications.
.VCn	One or more VSP Communications files if required. One file for each non-vital application that communicates with the Vital application. Shared by the Vital and non-vital applications.
.VNT	VSP network VSOE and DigiSAFE node declarations if required. Shared by Vital and Vital Comm applications.
.CW	VSOE and DigiSAFE link definition file if required. Shared with Vital Comm application if VSOE is used.
.VSL	VSOE, DigiSAFE, GTP and CRG messages file if required.
.PRM	Logic Parameters file (optional). CAAPE produces this file from graphics, but the logic parameters can also be put in the same file as the logic equations.
.VTL	Vital Logic file. The Vital logic should be the last section in the application data because the logic equations use variables which must be defined in previous sections.

6.5 SYSTEM COMM COMPILER INPUT FILES

The recommended INCLUDE file organization for the iVPI Comm (VSP board network) Compiler is shown in Table 6–2.

Table 6–2. Recommended INCLUDE File Organization for iVPI System Comm Compiler

File Type	Description
.VNT	VSP network VSOE and DigiSAFE node declarations if required. Shared by Vital and Vital Comm applications.
.CW	VSOE and DigiSAFE link definition file if required. Shared with Vital application.
.NVS	VSOE and DigiSAFE connection data if required: IP addresses and other network attributes of the VSOE and DigiSAFE nodes in a network.
.NMM	MMS connection data file if required (iVPICAA 610 and later): IP address and other network attributes of the MMS in a network.
.GW	Gateways file if required (iVPICAA 610 and later): IP address and other network attributes of the gateway devices used in a network.

6.6 NVSP COMPILER INPUT FILES

The recommended INCLUDE file organization for the NVSP (non-vital application) Compiler is shown in Table 6–3.

Table 6–3. Recommended INCLUDE File Organization For NVSP Compiler

File Type	Description
.HDW	Hardware file, shared by Vital and non-vital applications.
.VCn	One or more VSP Communications files if required. One file for each non-vital application that communicates with the Vital application. Shared by the Vital and non-vital applications.
.CSS	One or more Serial Communications files if required. Serial Communications data might also be shared between non-vital applications that communicate through serial ports, but this practice is not common. Be aware that the .COM extension was used in older CAA versions.
.NSS	Network Serial Communications file if required.
.NCW	NVSOE links definition file if required (iVPICAA 610 and later): network connections between NVSOE nodes.
.NNS	NVSOE connection data file if required (iVPICAA 610 and later): IP addresses and other network attributes of the NVSOE nodes in a network.
.NMM	MMS connection data file if required (iVPICAA 610 and later): IP address and other network attributes of the MMS in a network.
.GW	Gateways file if required (iVPICAA 610 and later): IP address and other network attributes of the gateway devices used in a network.
.LOG	Data Logger file if required.
.PRM	Logic Parameters file (optional). CAAPE produces this file from graphics, but the logic parameters can often be put in the same file as the non-vital logic. There is an exception to this: if the logic uses arrays, and array elements are used in other sections such as serial communications messages, then the file defining the logic parameters must be placed before the first section that uses an array element.
.NV	Non-vital logic file. The non-vital logic should be the last section in the application data because the logic equations use variables which must be defined in previous sections.

6.7 SHAREABLE NETWORK FILES

The following files could be shared among all the appropriate applications in a network of iVPI systems.

Table 6–4. Shareable Network Files

File Type	Description
.CW	VSOE and DigiSAFE link definitions. Used by Vital and Vital Comm Compilers. Could be shared among multiple systems as long as all VSOE and DigiSAFE names are unique throughout the network.
.NVS	VSOE and DigiSAFE connections data. Used by Vital Comm Compilers. Could be shared among multiple systems as long as all VSOE and DigiSAFE node names are unique throughout the network.
.NCW	NVSOE links definitions. Could be shared among multiple systems as long as all NVSOE node names are unique throughout the network.
.NNS	NVSOE connection data file. Could be shared among multiple systems as long as all NVSOE node names are unique throughout the network.
.NMM	MMS connection data file. Could be shared among multiple systems as long as all MMS names are unique throughout the network.
.GW	Gateways file if required. Could be shared among multiple systems as long as all gateway names are unique throughout the network.

SECTION 7 – HARDWARE FILE (.HDW)

The .HDW file defines the physical layout and properties of the system hardware module. A single hardware file is generally shared among all applications in the system

7.1 BASIC FILE FORMAT

The file consists of an iVPI System Module section and up to four optional iVPI Extender Module sections. The system module section starts with an iVPI SYSTEM MODULE SECTION header record; each extender module sections starts with an iVPI EXPANSION MODULE *n* SECTION header record, where n is the expansion module number is from 1 to 3.

Data for the module includes:

- The system or expansion module section header
- Hardware properties records (system module only)
- Module properties records
- Harness wiring subsection
- Board assignment subsection

7.2 HARDWARE PROPERTIES RECORDS

These are general hardware properties; they are placed only in the system module, but they apply to all modules. Hardware properties records should be placed just after the system module section header. All records are optional.

7.2.1 FLASH Records

This record is used by the non-vital compiler to set flash rates for non-vital outputs. Only one record is required if the default NVO flash rates are not acceptable.

The record format is:

FLASH = *flash-rate*

- *flash-rate* is the normal flash rate in flashes per second (for example, a 60 rate means 1/2 second off and 1/2 second on). Fast flash rate is always twice this entry.

For example:

```
FLASH = 50
```

7.2.2 PULSE, PULSE2 Records

These records are used by the non-vital compiler to set pulse widths for non-vital outputs. Only one instance of either record is required if a default pulse width is not acceptable.

The record format is:

PULSE = *pulse-width*

PULSE2 = *pulse2-width*

- *pulse-width* and *pulse2-width* are the PULSE and PULSE2 widths in milliseconds.

For example:

```
PULSE = 300
PULSE2 = 450
```

7.3 MODULE PROPERTIES RECORDS

These are general properties for an entire module. Module properties records should be placed before the board assignment subsection in the module. All record types are optional.

7.3.1 MODULE TYPE Records

This record specifies whether the module's motherboard is split, and which section of the module this system uses. Each compile refers to a single system; a separate set of input files and separate compiles is required for each system in a module.

The record formats are:

MODULE TYPE = **FULL**

MODULE TYPE = **SPLIT***(start-slot, num-slots)*

- FULL indicates that the motherboard is not split. This is the default condition which applies if the MODULE TYPE record does not exist.
- *start-slot* is the starting slot of the system within the module.
- *num-slots* is the number of slots available to the system within the module.

For example:

MODULE TYPE = SPLIT(5,4)

7.3.2 MODULE Records

This record gives the Alstom drawing number of this module.

The record format is:

MODULE = *module-num*

- *module-num* is the Alstom drawing number assigned to the module.

For example:

MODULE = 59473-557 GR 1

7.3.3 RACK Records

This record specifies the rack location of this module, and is used in the CAD annotation file (.CAD).

The record format is:

RACK = *rack-name*

- *rack-name* is the rack location of this module. Up to 20 alphanumeric characters may be used for this record.

For example:

RACK = 005

7.3.4 POSITION Records

This record specifies the position within the rack of this module, and is used in the CAD annotation file (.CAD).

The record format is:

POSITION = *position*

- *position* is the position of this module. Up to two alphanumeric characters may be used for this record.

For example:

POSITION = A

7.3.5 MODULE_PREFIX Records

This record describes the module type (for example: 3U, 9U), and is used in the CAD annotation file (.CAD).

The record format is:

MODULE_PREFIX = *prefix*

- *prefix* is the prefix to the module number. Up to five alphanumeric characters may be used for this record.

For example:

MODULE_PREFIX = SYS

MODULE_PREFIX = EXTD1

7.4 BOARD ASSIGNMENT SUBSECTION

This subsection starts with a BOARD ASSIGNMENT SUBSECTION header. It describes the slot assignments and properties of the boards in the module.

7.4.1 Board Assignment Records

7.4.1.1 SLOT Records

SLOT records assign a particular type of board to a slot in the module and gives basic information on the board. This is also the “header” for additional board information such as I/O groups: depending on the type of board, some of the other records described may follow to give additional data on the board.

The record format is:

SLOT *slot-num = board type, board data*

- *slot-num* is the module slot number, from 1 to 21.
- *board-type* is the board type name.
- *board-data* is additional board data including the board’s part number.

For example:

SLOT 01 = NVSP BOARD 1, 31166-428 GR 01

7.4.1.2 IONAME_n_PREFIX / IONAME_n_SUFFIX Records

IONAME_n_PREFIX, IONAME_n_SUFFIX records are used in the CAD annotation file (.CAD) to modify the I/O variable names for the board. A prefix and/or suffix may be added to the names for further description (for example, "+" or "P" for positive wires, "-" or "N" for negative wires).

These records are optional. If these records are used, they should follow the board's SLOT record and precede any I/O group or port records.

The record format for boards other than non-vital relay output is:

IONAME_+_PREFIX = *pos_prefix*

IONAME_+_SUFFIX = *pos_suffix*

IONAME_-_PREFIX = *neg_prefix*

IONAME_-_SUFFIX = *neg_suffix*

- *pos_prefix* is the prefix for positive wire I/O variable names.
- *pos_suffix* is the suffix for positive wire I/O variable names.
- *neg_prefix* is the prefix for negative wire I/O variable names.
- *neg_suffix* is the suffix for negative wire I/O variable names.

The record format for non-vital relay output boards is:

IONAME_F_PREFIX = *front_prefix*

IONAME_F_SUFFIX = *front_suffix*

IONAME_H_PREFIX = *heel_prefix*

IONAME_H_SUFFIX = *heel_suffix*

IONAME_B_PREFIX = *back_prefix*

IONAME_B_SUFFIX = *back_suffix*

- *front_prefix* is the prefix for front contact wire I/O variable names.
- *front_suffix* is the suffix for front contact wire I/O variable names.
- *heel_prefix* is the prefix for heel contact wire I/O variable names.
- *heel_suffix* is the suffix for heel contact wire I/O variable names.
- *back_prefix* is the prefix for back contact wire I/O variable names.
- *back_suffix* is the suffix for back contact wire I/O variable names.

Up to two characters may be used for each prefix record; only one character may be used for each suffix record.

For example:

IONAME_+_PREFIX = B_

IONAME_+_SUFFIX = +

IONAME_-_PREFIX = -

IONAME_-_SUFFIX = N_

7.4.1.3 GROUP Records

GROUP records identify the sources of positive and negative energies for a group of inputs or outputs. This record is required as input prior to specifying any I/O port data records for input or output ports within a group. It may be followed by group wirename CAD records, the I/O ports belonging to the group, and other data pertaining to the group.

The record format is:

GROUP *group* = *group-wiring*

- *group* is a one character group identifier.
- *group-wiring* consists of one or more wire names, depending on the type of board.

For example:

GROUP B = VB12-3,VN12-3

GROUP D = N12

GROUP A = NO WIRE, VN12-1

7.4.1.4 WIRENAME GROUP Records

WIRENAME_+_GROUP, WIRENAME_-_GROUP records are used in the CAD annotation file (.CAD) as wire names for the group positive and negative energies. These records are optional. One record may be used for each group energy wire.

The record format is:

WIRENAME_+_GROUP *group = positive-wire*

WIRENAME_-_GROUP *group = negative-wire*

- *group* is the one character group identifier (A, B, C, or D).
- *positive-wire* is the name for the positive group energy wire.
- *negative-wire* is the name for the negative group energy wire. Up to 16 characters may be used for each record.

For example:

WIRENAME_+_GROUP A = B12

WIRENAME_-_GROUP D = 1WAN12

7.4.1.5 Diagnostics Records

These records designate whether to enable board diagnostics for boards having onboard diagnostic circuitry. They can also designate a status variable to be used by the application logic to detect when board failure has occurred.

The record format is

DIAGNOSTICS = yesno [, diagnostic-var]

- *yesno* is YES or NO, indicating whether or not the option is used.
- *diagnostic-var* is the optional name of a variable which can be used to access board diagnostic results.

For example:

DIAGNOSTICS = NO

DIAGNOSTICS = YES, NVI1-DIAGS-RSLT

NOTICE

NVI/NVO boards with diagnostics must always have their diagnostic option set to YES. The VSP board controls the Health LEDs and must have diagnostic data from each NVI/NVO board.

7.4.1.6 External Power Supply Records

These records identify the external power supply wiring for non-vital output boards.

The record format is

PS supply = supply-wiring

- *supply* is the power supply designation, A for the first 16 ports of the NVO and B for the second 16 ports.
- *ps-wiring* are the wire names.

For example:

PS A = PSWIRE1, PSWIRE2

7.4.1.7 WIRENAME_n_PS Records

These records are used in the CAD annotation file (.CAD) to modify the external power supply names for the board. The record format is:

WIRENAME_+_PS *supply* = *name*

WIRENAME_-_PS *supply* = *name*

- *supply* is the power supply designation, A for the first 16 ports of the NVO and B for the second 16 ports.
- *name* is the annotation name.

For example:

WIRENAME_+_PS A = PSCADP

WIRENAME_-_PS A = PSCADN

7.4.1.8 I/O Port Records

I/O Port records define the use of the I/O ports on input and output boards. The I/O port identified on the data record is assigned to the currently defined I/O board. I/O ports not specifically identified and named are classified as PREWIRED SPARES for that board. This record is required as input for each I/O port to be used in an equation. Options for this record vary according to I/O board type.

The record format is:

port [(cof)] , name = wiring

- *port* is the port number.
- *cof* is the cycles of forgiveness for Vital input ports: 0CD, 1CD or 2CD.
- *name* is the name of the I/O variable.
- *wiring* is the port wiring.

For example:

1, 1EGE-SBO = 1EGE

13 (1CD) , 1EAHDGP-DI = 1EAHDGP,VN12-1

28, 3US-NVID = 3USPRP,3USPRN

7.4.1.9 I/O Port Mode Records

Depending on the board type, these records may follow an I/O port record to define the variables associated with available port modes such as Vital flash protection / filament checking.

The record format is:

port (mode) , name

- *port* is the port number.
- *mode* is the port mode.
- *name* is the associated variable name.

For example:

1 (FLASH) , 10ELFKE-FLASH

1 (COLD-LO-CK) , 1EAE-LO-CK

7.4.2 Board Type Specific Records

For details on hardware board application, see the iVPI Application Rules, section 5.1. Hardware Application Rules.

7.4.2.1 ACO Board

CAD Records: I/O and group wire names optional

Group Records: two groups of four ports each. Group has positive and negative wires.

Port Records: positive wire only. CK, FLASH and FLASH-STATE records available.

The ACO board slot assignment format is:

SLOT s = **ACO BOARD***(pair-slot) , part-num*

- *pair-slot* is the slot of the Vital output board paired with this one for addressing purposes, 0 if board is unpaired.
- *part-num* is the part number.

For example:

SLOT 4 = ACO BOARD(5), 31166-432GR2

GROUP A = B12,N12

1, 2EAG= N2EAGE

1 (CK), 2EAG-CK

1 (FLASH), 2EAG-FL

…

7.4.2.2 BEX Board

The BEX board slot assignment format is:

SLOT s = **BEX BOARD** *part-num*

- *part-num* is the part number.

7.4.2.3 Code Rate Generator (CRG) Board

CAD Records: I/O and group wire names optional

Group Records: two groups of four ports each. Group has positive and negative wires.

Port Records: positive and negative wires. Port names are for documentation only. The code and status bit names that are used in Vital application logic are defined in the CRG Communications Section.

The CRG board slot assignment format is:

SLOT s = **CRG BOARD** *crg-id , part-num*

- *crg-id* is the board number, 1 to max, corresponding to the board number in the CRG ID record in the VPC file.
- *part-num* is the part number.

For example:

SLOT 4 = CRG BOARD 1, 31166-459GR02

GROUP A = VCBA,VCNA

1 ,5WCT-CRG = 5WCT,N5WCT

2 ,4CCT-CRG = 4CCT,N4CCT

GROUP B = VCBB,VCNB

…

7.4.2.4 Genrakode Track Processor (GTP) Board

CAD Records: I/O wire names optional

The GTP board slot assignment format is:

SLOT s = **GTP BOARD** *gtp-id , part-num*

- *gtp-id* is the board number, 1 to max, corresponding to the board number in the GTP ID record in the VPC file.
- *part-num* is the part number.

For example:

SLOT 4 = GTP BOARD 1, 31166-434GR01

7.4.2.5 DBO Board

CAD Records: I/O and group wire names optional

Group Records: two groups of four ports each. Group has positive and negative wires.

Port Records: positive and negative wires; CK, FLASH, FLASH-STATE and ON-STATE records are available.

The DBO board slot assignment format is:

SLOT s = **DBO BOARD***(pair-slot) , part-num*

- *pair-slot* is the slot of the Vital output board paired with this one for addressing purposes, 0 if the board is unpaired.

For example:

SLOT 5 = DBO BOARD(4),59473-433GR01

GROUP A = B12, N12

1, 1NW-DBO= 1NW, 1RW

1 (CK), 1NW-CK

…

7.4.2.6 DI Board

CAD Records: I/O wire names optional

Port Records: positive and negative wires, cycles of forgiveness = 0 (0CD), 1 (1CD) or 2 (2CD).

The DI board slot assignment format is:

SLOT s = **DI BOARD** , *part-num*

For example:

SLOT 7 = DI BOARD, 59473-429GR01

1 (1CD), VRDFRNT-DI = VRDFRNT,N12

…

7.4.2.7 LDO Boards

CAD Records: I/O and group wire names optional

Group Records: two groups of four ports each. Group has positive and negative wires.

Port Records: positive wire only. ON, NONPROT, FLASH, FLASH-STATE, ALT-FLASH, ALT-FLASH-STATE, ON-STATE, HOT-LO-CK, COLD-LO-CK records available.

The LDO board slot assignment format is:

SLOT s = **LDO BOARD***(pair-slot) , part-num*

- *pair-slot* is the slot of the Vital output board paired with this one for addressing purposes, 0 if the board is unpaired.

For example:

SLOT 5 = LDO BOARD(4),31166-431GR01

GROUP A = B12,N12

1, 1NW-LDO= 1NW

1 (HOT-LO-CK), 1NW-HCK

1 (COLD-LO-CK), 1NW-CCK

…

7.4.2.8 Non-Vital Input Boards

CAD Records: I/O and group wire names optional

Diagnostics Records: Diagnostics must be enabled; diagnostic variable name optional.

Group Records: four groups of eight ports each. Group has positive and negative wires. Debounce record optional for each group.

Port Records: positive and negative wires.

The NVI board slot assignment format is:

SLOT s = **NVI BOARD***(nvsp-num) , part-num*

- *nvsp-num* is the number from 1 to max of the controlling NVSP board.
- *part-num* is the part number.

For example:

SLOT 10 = NVI BOARD(1), 31166-457GR01

DIAGNOSTICS = YES, NVI-TEST

GROUP A = B12,N12

DEBOUNCE = 25MSEC.

1,NVI-1 = NVI1,NVI1N

2,NVI-2 = NVI2,NVI2N

...

7.4.2.9 Non-Vital Output Boards

CAD Records: I/O, group and power supply wire names optional.

Diagnostics Records: Diagnostics must be enabled; diagnostic variable name optional.

External Power Supply Records: two groups of 16 ports, each with positive and negative wire names, only if the board has the external power supply feature.

Group Records: four groups of eight ports each. Group has a single wire name.

Port Records: front, back and heel wires for Groups A and C of NVO RELAY; front and heel for all others. NOWIRE can be used as a place-holder to indicate an unwired connection. FLASH, FLASH2, FLASHB, FLASH2B, PULSE and PULSE2 records available.

The NVO board slot assignment formats are:

SLOT s = **NVO RELAY BOARD***(nvsp-num) , part-num*

SLOT s = **NVO SSR BOARD***(nvsp-num) , part-num*

- *nvsp-num* is the number from 1 to max of the controlling NVSP board.
- *part-num* is the part number.

For example:

SLOT 11 = NVO RELAY BOARD(1), 31166-458GR01

DIAGNOSTICS = YES, NVO-TEST

PS A = PSAP,PSAN

PS B = PSBP,PSBN

GROUP A = GRPAP

1,NVORLY-1 = NVORLY1F,NVORLY1B,NVORLY1H

1 (FLASH), NVORLY-1FL

...

7.4.2.10 NVSP Board

CAD Records: I/O wire names optional

The NVSP board slot assignment format is:

SLOT s = **NVSP BOARD** *NVSP-id* , part-num [, **ETHERNET**, **CSEX MODE** (*mode*)] [**P3 ADAPTER**, *adapter-partnum]*

ETHERNET is an optional field specifying that the board is network-capable, i.e., that it contains network controller devices and can be used for network operations.

- *NVSP-id* is the board number, 1 to max, corresponding to the board number in the NVSP ID record in the VPC and CSI files.
- *mode* specifies whether the NVSP board's serial port data and control lines are hardware compatible with CSEX#.
- *adapter-partnum* is the part number of an optional interface adapter board which can be put onto the NVSP. It is contained in a P3 ADAPTER record that is placed on the next line after the NVSP slot assignment if the adapter exists.

For example:

SLOT 2 = NVSP BOARD 2, 31166-428GR01, ETHERNET, CSEX MODE(2)

P3 ADAPTER, 31166-475GR01

NOTICE

An NVSP board with a P3 Interface (P/N 31166-475GR01) requires two slots.

7.4.2.11 SBO Board

CAD Records: I/O and group wire names optional

Group Records: two groups of four ports each. Group has positive and negative wires.

Port Records: positive wires only; CK, FLASH, FLASH-STATE and ON-STATE records are available.

The SBO board slot assignment format is:

SLOT s = **SBO BOARD***(pair-slot) , part-num*

- *pair-slot* is the slot of the Vital output board paired with this one for addressing purposes, 0 if board is unpaired.

For example:

SLOT 5 = SBO BOARD(4),31166-430GR01

GROUP A = B12,N12

1, 1NW-SBO= 1NW

1 (CK), 1NW-CK

7.4.2.12 VSP Board

The VSP board slot assignment format is:

SLOT s = **VSP BOARD** , *part-num* [, **ETHERNET**]

ETHERNET is an optional field specifying that the board is network-capable, i.e., it contains network controller devices and can be used for network operations.

For example:

SLOT 1 = VSP BOARD,31166-427GR01, ETHERNET

SECTION 8 – VSP-NVSP COMMUNICATIONS FILE (.VCN)

The .VCn file describes the DPRAM communications between the Vital application on the VSP board and one of the non-vital applications on an NVSP board. One .VCn file is included in the .VPC file of the Vital application and the .CSI file of the non-vital application for each NVSP board that communicates with the VSP.

For example, if there were two NVSP boards communicating with the VSP board, the include files might be used as follows:

SYS_VITAL.VPC File

```
INCLUDE SYS.VC1
INCLUDE SYS.VC2
```

SYS_NV1.CSI File (NVSP board #1)

```
INCLUDE SYS.VC1
```

SYS_NV2.CSI File (NVSP board #2)

```
INCLUDE SYS.VC2
```

The file contains the VSP COMMUNICATIONS section, which consists of a section header record followed by a source message and a destination message. “Source” and “Destination” are from the point of view of the NVSP board: the Source message is the one sent from the NVSP board to the VSP. Each message consists of a SOURCE or a DESTINATION record followed by a number of message bits.

The .VCn file format is:

VSP COMMUNICATIONS SECTION

SOURCE = NVSP-name

1 = *NVSP-VSP-bit-1*

…

N = *NVSP-VSP-bit-N*

DESTINATION = *NVSP-name*

1 = *VSP-NVSP-bit-1*

…

N = *VSP-NVSP-bit-N*

- *NVSP-name* is the name of the NVSP board identified in an NVSP ID record in the .VPC and .CSI files.
- *NVSP-VSP-bit* is the name of a variable being sent from the NVSP board to the VSP board.
- *VSP-NVSP-bit* is the name of a variable being sent from the VSP board to the NVSP board.

Message bits are numbered from 1 to the message size. A bit name of PERMZERO is allowed to indicate that a bit is unused or always has a value of False.

For example:

```
VSP COMMUNICATIONS SECTION


* NVSP-TO-VSP MESSAGE
SOURCE = SAMPLE FILE NVSP
1 = 1GZ
2 = F1GE-DISP
3 = 1AS
4 = 1AE
5 = PERMZERO
6 = PERMZERO
7 = PERMZERO
8 = PERMZERO


* VSP-TO-NVSP MESSAGE
DESTINATION = SAMPLE FILE NVSP
1 = 1HP
2 = 1FS
3 = 1LO
4 = 1OSTP
5 = 1TE
6 = A1H-CK
7 = A1D-CK
8 = PERMZERO
```

SECTION 9 – COMPILER FILES

This section discusses the files used specifically by the iVPI Compiler. The Hardware and VSP-NVSP Communications files are shared with the NVSP Compiler and have been described in an earlier section of the manual.

9.1 MAIN COMPILER FILE (.VPC)

This file contains documentation records for the Vital application and INCLUDE records referencing additional files with the rest of the Vital application data. The INCLUDE records usually follow any documentation records in the file.

9.1.1 Revision History

Application revision history can be stored as special comments in the main (.VPC or .CSI) compiler input file. These comments are read by the Relay Equivalent Drawing Program.

The .VPC file format is:

```
*%REV id : date : author : summary
*+ details
```

- *id* is a unique version identifier.
- *date* is the revision date.
- *author* is the revision author.
- *summary* is a short description of the change.
- *details* are optional records detailing the change. Details records must follow the revision header record, but other comments can be placed between revisions.

For example:

```
**********************************************
*%REV A : 03/04/05 : JRM : INITIAL
**********************************************
*%REV B : 05/04/05 : JRM : RELEASE
*+ ADD SECOND VSC INTERFACE
**********************************************
```

9.1.2 File Records

9.1.2.1 COMPILER RUN CONTROLS Records

This record comprises a list of override commands to control the processing of the iVPI compiler program. If used, this record must be the first non-comment data record in the compiler input file. This record is ordinarily not needed; run controls can now be specified through CAAPE's user interface.

The record format is:

COMPILER RUN CONTROLS = *data-1, data-2, data-n*

- *data-1* though *data-n* are a list of compiler run controls chosen from the Table 91. If the list exceeds the limit of one data record, it may be continued in subsequent data records provided the last non-blank character on the preceding data record is a comma (,). The order of the output reports is predetermined by the compiler. These run controls only indicate whether or not the report is to be generated, and do not define order. Commands are processed in the order entered on the data record. It is possible to both select a given command and then accidentally cancel it if the command is specified in both its positive and negative sense; for example, XREF,....., NO XREF.

For example:

```
COMPILER RUN CONTROLS = LIST ALL,NO XREF,NO WIRE
COMPILER RUN CONTROLS = NO LIST ALL,PROM,BOARD,PLUG,WIRE
```

NOTICE

When performing an initial build, the last-installed CAA version is automatically used to do the compile. This is generally the version that uses the latest available Vital system software. If a different CAA version is desired, or to use non-default compiler options, do a Make Files first to create the applications and then go to the ***FileView*** and set the run controls for each application that was created. See P2512A CAAPE User Manual for more details.

Table 9–1 lists the run control commands along with their definitions.

Table 9–1. iVPI Compiler Run Control Commands

Command	Action
LIST ALL NO LIST ALL	Permits the user to enable or disable all of the Compiler output reports. This command may be specified as the general condition for all output reports, and then overridden for specific reports by supplying the proper run control. The NO LIST ALL run control is the default option for the Compiler. These controls are not affected by this command and must be uniquely specified to be enabled: PROM, I/OLABEL, ADV, NO COMP, GRAPHISM.
BOOLEAN NO BOOLEAN	Controls the Boolean equation section report. This report lists the user's Boolean equation section data records plus all iVPI Compiler-generated equations. Input equations, generated equations, and the equations retrieved from libraries are shown in expanded sum-of-products form.
BOARD NO BOARD	Controls the board report for the modules. This report fully describes the use of each board. The information given includes naming and wiring information for I/O ports, address assignments for all boards, unassigned and prewired spares on I/O boards, and wiring and switch settings for the system boards. Also controls the creation of the CAD annotation file (.CAD), which contains most of the board report information in a tabular form.
PARAM NO PARAM	Controls the listing of the parameter name report. This report lists parameter and I/O names, their internal address assignments and the 32-bit codewords they have been assigned.
BDEDGE NO BDEDGE	Controls the board edge connector report. This report fully describes the wiring assignments of each board connector. It also shows the location of UNUSED and NO WIRE pins.
WIRE NO WIRE	Controls the wire table report. This report provides a complete wire table for a module at a time.
XREF NO XREF	Controls the listing of the parameter name and I/O cross reference report. This report provides a listing in alphanumeric character order of every parameter, input and output name used in the compilation, and its type. In addition, the report shows the use of the names in the expressions.

Table 9–1. iVPI Compiler Run Control Commands *(Cont.)*

Command	Action
INDEX NO INDEX	Controls the listing of the index report. This report provides an index to the Compiler listing by page number, giving the starting page number of each report. Also, if the user has chosen APPLICATION and LOCATION records within the Boolean equation section, the numbers of the pages on which these records appear are also listed.
PROM	Controls the generation of an output file of iVPI PROM code. No listing report is produced. This run control must be uniquely specified, it is not enabled by the LIST ALL run control.
I/O LABEL	Controls the generation of the I/O Label file, which is needed to create the front cover labels of input and output names for iVPI modules. This run control must be uniquely specified, it is not enabled by the LIST ALL run control.
NO COMP	Controls the compilation of the Boolean equation section. This command prevents the Compiler from processing the Boolean equation section of the input file. It saves computer time and expense when producing reports and output files that are not dependent upon the Boolean equation section data (for example, the I/O label generation file). This run control must be uniquely specified, it is not enabled by the LIST ALL run control.
ADV	Initiates symbol table file creation, which is used by the iVPI ADV Program. The symbol table file contains the user-assigned name of each parameter and its internal address. It also causes the compiler to produce an expression result PD sum, for comparison with the corresponding ADV report. This run control must be uniquely specified, it is not enabled by the LIST ALL run control.
GRAPHSIM	Causes a Graphical Simulator data file to be created. This run control must be uniquely specified, it is not enabled by the LIST ALL run control.

9.1.2.2 APPLICATION PROGRAM NUMBER Records

This record provides the top-level archive name or part number for the entire iVPI system. In addition to the name or number, the current revision letter of the file and the initials of the person responsible for the latest updates can be included on this record. This record is optional.

The record format is:

APPLICATION PROGRAM NUMBER = program-num, REV.rev

- *program-num* is an archive name of up to 13 characters or an Alstom drawing number.
- *rev* is whatever revision information is needed to document the system. For instance the current revision date, revision letter, and the initials of the person responsible for making the latest updates could be included in this field. A maximum of 21 characters are saved for this field, which is output in the headings of all documentation generated by the compiler.

For example:

APPLICATION PROGRAM NUMBER = 32917-001 GR 00, REV. C, JKL

APPLICATION PROGRAM NUMBER = ARCHIVENAME, REV. D, JRM

9.1.2.3 VSP PROGRAM NUMBER Records

This record provides the archive name or part number of this VSP application program. This means the number assigned to the specific VSP application code, not the system software drawing number (see the SYSTEM SOFTWARE Record description). In addition to the name or number, the current revision letter of the file and the initials of the person responsible for the latest updates should be included on this record.

The data on this record is placed in the heading of every page of the compiler output reports. It is also placed on the module I/O labels for VSP boards.

The record format is:

VSP PROGRAM NUMBER = program-num, **REV**.rev

- *program-num* is an archive name of up to 13 characters or an Alstom drawing number assigned to the program.
- *rev* is whatever revision information is needed to document the system. For instance the current revision date, revision letter, and the initials of the person responsible for making the latest updates could be included in this field. A maximum of 21 characters are saved for this field, which is output in the headings of all documentation generated by the compiler.

For example:

VSP PROGRAM NUMBER = 32917-001 GR 00, REV. C, JKL

VSP PROGRAM NUMBER = IVPIPROGNAME, REV. D, JRM

9.1.2.4 COPYRIGHT YEAR Records

This record denotes the year of copyrighting of the VSP Program. This means the copyright of the application code (VSP PROGRAM NUMBER), not the system software copyright date.

The record format is:

COPYRIGHT YEAR = year

- *year* is the 4-digit date of software copyright.

For example:

COPYRIGHT YEAR = 1985

9.1.2.5 SYSTEM SOFTWARE Records

This record contains the Alstom drawing number of the VSP system software, the routines that run using the application data generated by the Compiler Program. This is not the drawing number assigned to the specific application code (see the VSP PROGRAM NUMBER Record description).

This record is required for an EPROM code data file to be generated. If the system software drawing number does not match the operating software, Vital energy is not supplied for the outputs and the iVPI system cannot operate. In addition, the Compiler program verifies that the system software is a version that is supported, and alerts the user if it is not.

The record format is:

SYSTEM SOFTWARE = *partnum*

- *partnum* is the Alstom drawing number assigned to the program. A revision letter may be required depending on the CAA version.

For example:

SYSTEM SOFTWARE = 40025-413 GR 00, REV.B

9.1.2.6 CONTRACT NUMBER Record

This record identifies the contract number(s) for which this VSP Program is being provided.

The record format is:

CONTRACT NUMBER = *contract-num*

- *contract-num* is the contract number(s), maximum of 40 characters in length.

For example:

CONTRACT NUMBER = 91-79897

CONTRACT NUMBER = 91-79400,91-80500

CONTRACT NUMBER = PD221

9.1.2.7 CONTRACT NAME Records

This record identifies the contract name(s) for which this VSP Program is being provided.

The record format is:

CONTRACT NAME = *contract-name*

- *contract-name* is the contract name(s), a maximum of 40 characters in length.

For example:

CONTRACT NAME = KENTON AVE. CHICAGO

9.1.2.8 CUSTOMER NAME Record

This record identifies the customer name for whom this VSP Program is provided.

The record format is:

CUSTOMER NAME = *cust-name*

- *cust-name* is the customer's name, a maximum of 40 characters in length.

For example:

CUSTOMER NAME = CONRAIL

9.1.2.9 EQUIPMENT LOCATION Record

This record identifies the physical location of the equipment for which this VSP Program is being provided.

The record format is:

EQUIPMENT LOCATION = *location*

- *location* is the physical location of the iVPI module(s) in which this program is located, a maximum of 40 characters in length.

For example:

EQUIPMENT LOCATION = OLD RELAY ROOM, RACK A3

9.1.2.10 DESIGNER Records

This record identifies the name(s) of the individual(s) responsible for the design of this VSP Program.

The record format is:

DESIGNER = *name*

- *name* is the name(s) of the designer(s), a maximum of 40 characters in length.

For example:

DESIGNER = JOHN Q. DESIGNER

9.1.2.11 CHECKER Records

This record identifies the name(s) of the person(s) responsible for checking all aspects of the iVPI compiler input file, including the Boolean equations, the I/O, and the parameter definitions.

The record format is:

CHECKER = *name*

- *name* is the name(s) of the individual(s) responsible for checking these equations, a maximum of 40 characters in length.

For example:

CHECKER = JUNE D. CHECKER

9.1.2.12 VSP ID Records

VSP boards only. This optional record provides a user-readable name for the VSP board; the name can be used to identify the board on a network.

The record format is:

VSP ID = *board-name*

- *board-name* is a board name of up to 40 characters.

For example:

VSP ID = IVMAL1A

9.1.2.13 VITAL OUTPUT FLASHING Records

This record specifies that the system contains one or more lamp drive output boards, and that some of the lamps are required to flash. If the system has to be protected against inadvertent flashing, then this record must be provided. This record is optional, and should be used only if there are lamp drive outputs which flash under certain conditions. If this record is not present, data records pertaining to Vital flashing are not recognized. (VITAL OUTPUT FLASHING = NO is assumed.)

The record format is:

VITAL OUTPUT FLASHING = *yesno*

- *yesno* is either 'YES' if Vital output flashing is present or 'NO' if there are no lamp drive output boards or no lamp drive outputs that flash.

For example:

VITAL OUTPUT FLASHING = YES

VITAL OUTPUT FLASHING = NO

9.1.2.14 RESTRICTIVE FLASHING ASPECTS Records

This record specifies whether a flashing signal is more restrictive than a steady one; this condition is prevalent in European applications. The record should be included only if Vital flashing was selected, and should be the first non-comment record after Vital OUTPUT FLASHING.

Two types of flash protection are available. In the first, a flash state parameter must be declared for each protected port. This parameter is true only if no current flows through the port during the last half of the system cycle. In the second type, correct recheck checkword production requires no current through flashing ports during the last half of the cycle. An incorrect steady on condition in any port causes the VRD to drop.

This record is optional unless restrictive flashing aspects are required; permissive flashing is the default condition.

The record format is:

RESTRICTIVE FLASHING ASPECTS = *mode*

- *mode* is YES if output flashing is more restrictive than output on steady and state parameters is used, CHECKWORD if flashing is more restrictive and recheck checkword protection is used, or NO if flashing is not more restrictive.

For example:

RESTRICTIVE FLASHING ASPECTS = YES
RESTRICTIVE FLASHING ASPECTS = CHECKWORD
RESTRICTIVE FLASHING ASPECTS = NO

9.1.2.15 SOFTWARE REVISION / SITE ID Records

⚠WARNING

SOFTWARE REVISION CONTROL MUST BE MAINTAINED

Failure to properly version control iVPI system software and application data can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that strict revision control of the iVPI application data and system software be maintained so that the expected configuration in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

UNIQUE SITE ID CONTROL MUST BE MAINTAINED

Failure to properly assign, maintain and control unique Site IDs for iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that strict control of the Site IDs be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

ACCURATE SOFTWARE REVISION ID CONTROL MUST BE MAINTAINED

Failure to update and maintain the Software Revision IDs for **every** software change made to the application data and/or system software (even a re-compile done with no software changes) jeopardizes proper software revision control and can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that Software Revision IDs be changed with **every** software change, even a re-compile of unchanged software. Software Revision IDs shall be maintained so that software and application revision control is maintained and the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

The SOFTWARE REVISION ID record associates a set of application data with a specific iVPI system location. It is used for configuration control for iVPI systems containing VSP boards. Its value should be changed for each new installed revision of the application data. This record or the equivalent SYSTEM ID record is required; if both records are omitted, then no application data is generated.

The optional SOFTWARE SITE ID record can be used in combination with the SOFTWARE REVISION ID to identify a particular iVPI system in a location with multiple systems. Its value should be changed for each iVPI system in the location.

These records correspond to a wiring configuration on the iVPI module connector for the VSP board, and a mismatch (for example, plugging the incorrect board into the module) causes the system to NOT operate: Vital energy is not supplied to the outputs. An optional output of the Compiler Program is the wire list for configuring the iVPI module to match the provided ID numbers.

The record format is:

SOFTWARE REVISION ID = *rev-id*

SOFTWARE SITE ID = *site-id*

- *rev-id* is a 2-digit decimal number, 01-62.
- *site-id* is a 4-digit decimal number, 01-1022.

For example:

SOFTWARE REVISION ID = 24

SOFTWARE SITE ID = 1014

9.1.2.16 SYSTEM ID Records

WARNING

UNIQUE SYSTEM ID CONTROL MUST BE MAINTAINED

Failure to properly assign, maintain and control a unique System ID for each iVPI system within the entire train control system can result in unintended consequences including death or serious injury due to train collision or derailment.

Alstom strongly recommends that strict control of the System IDs be maintained so that the expected configuration of all iVPIs within the entire train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system, which deviate from Alstom's originally, delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

The SYSTEM ID record can be used instead of the SOFTWARE REVISION ID and SOFTWARE SITE ID records to associate a set of application data with a specific iVPI system location. Its value should be changed for each new installed revision of the application data. This record or the equivalent SOFTWARE REVISION ID record is required; if both records are omitted, no application data is generated.

This record corresponds to a special wiring configuration on the iVPI module connector for the VSP, and a mismatch (for example, plugging the incorrect board into the module) causes the system to NOT operate. Using this record causes the compiler to produce a special module wiring configuration that supports WAGO or punch block selection of the ID data.

The System ID can be entered in decimal or hexadecimal format.

The record format is:

SYSTEM ID = *sys-id*

- *sys-id* is a 5-digit decimal number from 1 to 65534, or a hexadecimal number from 0X0000 to 0XFFFE (the "0X" prefix indicates that the data is in hexadecimal format).

For example:

SYSTEM ID = 256

SYSTEM ID = 0X0100

9.1.2.17 NVSP ID Records

These records assign unique names to NVSP boards so communication messages can be assigned to a source NVSP and/or a destination NVSP. One of these records is required as input for each NVSP board in a system. The information in these records is used in the VSP Communications section(s).

The record format is:

NVSP *NVSP-id* **ID** = *NVSP-name*

- *NVSP-id* is the number assigned to the NVSP board on the hardware slot assignment record which must be 1, 2, 3, or 4.
- *NVSP-name* is a 40-character name given to the board. This name should describe the function of the board or the board's location, if possible. The names for each NVSP board at an installation should be unique even if the NVSP boards are in different modules connected serially.

For example:

NVSP 1 ID = LOCAL CONTROL PANEL DRIVER

NVSP 2 ID = MAIN CODE UNIT

NVSP 3 ID = STANDBY CODE UNIT

9.1.2.18 NVSP PROGRAM NUMBER Records

This record provides the archive name or Alstom drawing number of the application logic for each NVSP board. This means the drawing number assigned to the specific application code, NOT the system software drawing number. In addition to the name or drawing number, the current revision letter of the file and the initials of the person responsible for the latest updates can be included on this record.

The record format is:

NVSP *NVSP-id* **PROGRAM NUMBER** = *program-num , rev*

- *NVSP-id* is the number assigned to the NVSP board on the slot assignment record that must be 1, 2, 3, or 4.
- *program-num* is an archive name of up to 13 characters or an Alstom drawing number assigned to the program.
- *rev* is whatever revision information is needed to document the system. For instance, the current revision date, revision letter and the initials of the person responsible for making the latest updates could be included in this field.

For example:

NVSP 1 PROGRAM NUMBER = 32917-037 GR 00, REV. A, MNO

NVSP 1 PROGRAM NUMBER = NVSPAME, REV. A, MNO

9.1.2.19 GTP ID Records

These records give each GTP board in the system an identifying name to be used when specifying transmitted or received messages. One of these records is required for each GTP board in the system.

The record format is:

GTP *gtp-id* **ID** = *gtp-name*

- *gtp-id* is a number from 1 to 8 assigned to the GTP board on the slot assignment record.
- *gtp-name* is a name of up to 40 characters length. This name identifies the specific GTP board; it must be unique for all GTP boards in the system.

For example:

GTP 1 ID = SYSTEM A BOARD 1

GTP 2 ID = MAIN-WEST LINK, WEST END

9.1.2.20 GTP EXPORT FILE Records

This optional record identifies the name of an export file to be output by the VSP Compiler and read by the GTP CAA to provide shared information.

The record format is:

GTP *gtp-id* **EXPORT FILE** = *file-name*

- *gtp-id* is the number from 1 to 8 assigned to the GTP board on the slot assignment record.
- *file-name* is the base name of the export file. An extension of iGT is added when the export file is created.

For example:

```
**** full file name is GTP1EXPORT.iGT
GTP 1 EXPORT FILE = GTP1EXPORT
```

9.1.2.21 CRG ID Records

These records give each CRG board in the system an identifying name to be used when specifying transmitted or received messages. One of these records is required for each CRG board in the system.

The record format is:

CRG *crg-id* **ID** = *crg-name*

- *crg-id* is a number from 1 to 3 assigned to the CRG board on the slot assignment record.
- *crg-name* is a name of up to 40 characters length. This name identifies the specific CRG board; it must be unique for all CRG boards in the system.

For example:

```
CRG 1 ID = CRG BOARD 1
```

9.1.2.22 CRG SOFTWARE Records

These records contain the Alstom drawing number of the system software for each CRG board in the system. One of these records is required for each CRG board in the system.

The record format is:

CRG *crg-id* **SOFTWARE** = *sw-num*

- *crg-id* is a number from 1 to 3 assigned to the CRG board on the slot assignment record.
- *sw-num* is the Alstom drawing number of the system software for that board.

For example:

CRG 1 SOFTWARE = 33036-016 GR00

9.1.2.23 PASSWORD Records

This record contains the password used for gaining access to and changing the FSSVT Timers. The AlsDload utility requires the User to provide this password in order to change the values of the FSSVT Timers outside of the CAAPE/CAA environment.

The record format is:

FSSVT PASSWORD = *password*

- *Password* is a value containing a minimum of 6 letters and numbers and a maximum of 16 (not case sensitive, no spaces allowed).

For example:

FSSVT PASSWORD = keylargo

9.1.2.24 iVPI ID Records

This record contains the equipment ID of this iVPI that will be communicating with DigiSAFE Zone Controllers.

The record format is:

IVPI ID = *equipment ID of iVPI*

For example:

IVPI ID = 33452

NOTICE

DigiSAFE communications is only available in certain CAAs.

9.1.2.25 LIBRARY PATH Records

One of these optional records can be placed anywhere in the documentation section to indicate the path and/or library file name to be used when accessing library data in the Vital logic. It should be used only when a single path is being searched or a single library file is used.

The record format is:

LIBRARY PATH = *path [\ filename]*

- *path* is the path to be searched.
- *filename* is an optional file name, which can be included if a single library file is to be accessed. If no extension is given, '.LIB' is assumed.

For example:

LIBRARY FILE = D;\LIBFILES\CTA\NORTH.LIB

LIBRARY FILE = LIBFILE

9.1.2.26 INCLUDE Records

These records are used to identify additional files for the Compiler to read. INCLUDE records are generally the last non-comment records in the file.

The record format is:

INCLUDE = *file*

- *file* is the path/name of the file to be read. If there is no directory information, the Compiler looks for the file in the same directory as the main file. If the file name has embedded spaces, it must be enclosed in quotes.

For example:

INCLUDE MYAPP.HDW

INCLUDE “NVSP2 APP.VC1”

9.2 VSOE AND DIGISAFE NODES DECLARATION FILE (.VNT)

This file identifies the names and types of all VSOE and DigiSAFE nodes in a system. This file is required if VSOE or DigiSAFE exist in the system. It is shared with the Vital Comm application.

The file consists of a section header followed by a series of VSOE and/or DigiSAFE declarations. All node names must be unique.

The file format is

VSOE NODES SECTION

node-declaration-1

node-declaration-2

…

The format of each node declaration is:

VSOE *vsoe-num* **ID** = **TYPE**(*vsoe-type*), *vsoe-name*

or

DIGISAFE *DigiSAFE-num* **ID** = **TYPE***(DigiSAFE-type), DigiSAFE-name*,**ZCID=**zc-num

- *vsoe-num* is a number from 1 to N, where N is the number of nodes in the system. These numbers must be sequential.
- *vsoe-type* is the type of the node, identifying how the node is used in the system. Available types are:
  - *PEER* – the node has a bidirectional link to a remote node in another system.
- *vsoe-name* is a unique name for the node. This name is used to identify the node in all other files that refer to it.
- *DigiSAFE-num* is a number from 1 to N, where N is the number of nodes in the system. These numbers must be sequential.
- *DigiSAFE-type* is the type of the node, identifying how the node is used in the system. Available types are:
  - *DS_PEER* – the node has a bidirectional link to a remote node in another system.
- *DigiSAFE-name* is a unique name for the node. This name is used to identify the node in all other files that refer to it.
- Zc-num is the unique identifying equipment id for the Zone Controller this node will be communicating with.

For example:

```
DIGISAFE NODES SECTION

DIGISAFE 1 ID = TYPE(DS_PEER), BA4-ZC1,ZCID=1234
DIGISAFE 2 ID = TYPE(DS_PEER), BA4-ZC2,ZCID=5678
```

NOTICE

DigiSAFE communications is only available in certain CAAs.

9.3 VITAL SERIAL LINK DEFINITION FILE (.CW)

⚠WARNING

VITAL COMMUNICATIONS REQUIRE UNIQUE LINK AND BLOCK SETTINGS

Failure to properly assign, maintain and control unique Link and Block settings for Vital communications within iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

The message link and block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link and Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

For train control systems designed by Alstom, the transit or railroad authority shall be solely responsible for any modifications whatsoever to the train control system which deviate from Alstom's originally delivered design, and any consequences to the system's safety integrity and performance as a result of such modifications. Alstom assumes no responsibility or liability for any modifications to the train control system or for the safe performance of the train control system once Alstom's originally delivered design has been modified.

For train control systems not designed by Alstom, the transit or railroad authority shall be solely responsible for the design of the train control system, and any consequences to the system's safety integrity and performance as a result of such designs. Alstom assumes no responsibility or liability for any designs or for the safe performance of the train control system.

This file is a map of all VSOE (Vital Serial over Ethernet) and DigiSAFE links connecting this system and any others. The file is required if any such links exist in this system. It may be shared among multiple iVPI systems. If VSOE or DigiSAFE exists, the file is shared between the Vital and the Vital Comm applications.

The file consists of a section header followed by one on more link definitions. Each link definition has Source and/or Destination VSOE and DigiSAFE nodes. The VSOE and DigiSAFE nodes are identified by names matching those in the VSOE ID and DIGISAFE ID records in the .VNT file. If none of the sources or destinations in a link definition appears in a .VNT or .VCC file, the link definition is ignored by the Compiler.

The file format is:

VSC LINK DEFINITION SECTION

link-definition-1

link-definition-2

…

⚠WARNING

LINK-NUM ASSIGNMENT CRITICAL

Correct assignment of this *link-num* number is critical for system safety. The message link values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Link settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

Failure to properly assign, maintain and control unique Link settings for iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

BLOCK-NUM ASSIGNMENT CRITICAL

Correct assignment of this *block-num* number is critical for system safety. The message block values must be assigned such that the combination of these values is unique throughout the network.

Alstom strongly recommends that strict control of the Block settings be maintained so that the expected configuration of all iVPIs in the train control system is the actual installed configuration.

Failure to properly assign, maintain and control unique Block settings for iVPI systems can result in unintended consequences including death or serious injury due to train collision or derailment.

THE FORMAT OF EACH VSOE-TO-VSOE LINK DEFINITION IS:

VSOE LINK *link-num* = **LENGTH***(length)*,**BLOCK***(block-num)* [,**REDUNDANT**]

SOURCE = *source-vsoe-name*

DESTINATION = *dest-vsoe-name-1* [, **GATEWAY***(gateway-name)*]

…

- *link-num* is a number from 1 to maximum, unique for all VSC links in all interconnected systems for this application. The maximum link number is 200.
- *length* is a number from 1 to 450 giving the number of message parameters for this link.
- *block-num* is a codeword block number from 0.01 to 17.00 defining a set of codewords unique for all Vital serial links within this particular system. See the section on hardware application rules for information on VSC link and block assignment.
- *source-vsoe-name* is the name of the transmitting VSOE node. If this name matches one of the names in a VSOE ID record in the VNT file, that node is identified as the transmitting node.
- *dest-vsoe-name* is the name of a receiving VSOE node. If this name matches one of the names in a VSOE ID record in the VNT file, that node is identified as the receiving node. REDUNDANT indicates that the link is path-redundant. This field is available only for CAA versions with subnets/redundancy.
- *gateway-name* is the user name(s) of the gateways to be used if the destination VSOE is on a different subnet. This field is available only for CAA versions with subnets/redundancy. A corresponding GATEWAY record must be found in the GATEWAYS SECTION. Two names separated by a comma are required if the link is redundant; if only one path of a redundant link requires a gateway, the gateway name for the other path should be NONE.

The format of each DigiSAFE link definition is:

DIGISAFE MSG *msg-num* = **LENGTH***(length)*

[,**REDUNDANT**]

SOURCE = *source-DigiSAFE-name*

DESTINATION = *dest-DigiSAFE-name-1* [, GATEWAY*(gateway-name)*]

DESTINATION = *dest-DigiSAFE-name-1* [, GATEWAY*(gateway-name)*]

…

- *msg-num* is a number from 201 to maximum of 206. The maximum msg number is 206. A maximum of 3 Zone Controllers can be defined and each one will take a msg-num for message out and one for message in.
- *length* is set to 160 - giving the number of message parameters for this message. DigiSAFE messages must be 160 bits in length.
- *source-DigiSAFE-name* is the name of the transmitting DigiSAFE node. If this name matches one of the names in a DIGISAFE ID record in the VNT file, that node is identified as the transmitting node.
- *dest-DigiSAFE-name* is the name of a receiving DigiSAFE node. If this name matches one of the names in a DIGISAFE ID record in the VNT file, that node is identified as the receiving node. There must be two destinations defined. iVPI must communicate redundantly with the two units inside the zone controller. There is a Unit A and a Unit B.
- REDUNDANT indicates that the link is path-redundant. This field is available only for CAA versions with subnets/redundancy but is required for DigiSAFE communications.
- *gateway-name* is the user name(s) of the gateways to be used if the destination Zone Controller is on a different subnet. This field is available only for CAA versions with subnets/redundancy. A corresponding GATEWAY record must be found in the GATEWAYS SECTION. Two names separated by a comma are required if the link is redundant.

For example:

```
VSC LINK DEFINITION SECTION

VSOE LINK 14 = LENGTH(50), BLOCK(4)
SOURCE = BA4-IVM
DESTINATION = IVM1
DESTINATION = IVM2

VSOE LINK 15 = LENGTH(24), BLOCK(4), REDUNDANT
SOURCE = BA4-IVM-RED
DESTINATION = IVM1-RED
DESTINATION = IVM2-RED, GATEWAY(GW-2,NONE)

DIGISAFE MSG 201 = LENGTH(160),REDUNDANT
SOURCE = REM_ZC_01A
SOURCE = REM_ZC_01B
DESTINATION = DS_01

DIGISAFE MSG 202 = LENGTH(160),REDUNDANT
SOURCE = DS_01
DESTINATION = REM_ZC_01A, GATEWAY(GW-01,GW-01)
DESTINATION = REM_ZC_01B, GATEWAY(GW-01,GW-01)
```

NOTICE

DigiSAFE communications is only available in certain CAAs.

9.4 VSOE / GTP / CRG / DIGISAFE MESSAGES FILE (.VSL)

This file defines the VSOE-to-VSOE, DigiSAFE-to-DigiSAFE, GTP and CRG messages for this system. It is required if any of these messages exist. The file may contain Network VSC, GTP Communications and CRG Communications sections.

9.4.1 Network VSC Section

This section consists of a section header followed by one or more VSOE-to-VSOE or DigiSAFE-to-DigiSAFE messages. Each VSOE node in the system typically has one transmit and/or one receive message. Each VSOE message must correspond to one of the VSOE link definitions in the Vital serial link definition file. Each DigiSAFE node in the system has one transmit and one receive message. Each DigiSAFE message must correspond to one of the DigiSAFE link definitions in the Vital serial link definition file.

The data format is:

NETWORK VSC SECTION

vsoe-message-1

vsoe-message-2

…

The format of each VSOE message is:

SOURCE = *source-vsoe-name* -or- **DESTINATION** = *dest-vsoe-name*

1 = vsoe-bit-1

2 = vsoe-bit-2

…

- *source-vsoe-name* is the name of the transmitting VSOE node. If this name matches one of the names in a VSOE ID record in the VNT file, that node is identified as the transmitting node.
- *dest-vsoe-name* is the name of the receiving VSOE node. If this name matches one of the names in a VSOE ID record in the VNT file, that node is identified as the receiving node. The messages in this section typically have either a source or a destination name.
- *vsoe-bit-1* through *vsoe-bit-N* are the message bits. Message bit numbers can range from 1 to 200 for VSOE-to-VSOE messages; the maximum bit number in the message must match the message length for this link in the Vital serial link definition file. Bit names are the variables associated with the bit. PERMZERO is allowed to indicate that a transmitted bit's value is always False.

9.4.2 DigiSAFE Section

This section consists of a section header followed by one or more DigiSAFE messages. Each DigiSAFE node in the system has one transmit and one receive message. Each message must correspond to one of the DigiSAFE link definitions in the Vital serial link definition file.

The data format is:

DIGISAFE SECTION

DigiSAFE-message-1

DigiSAFE-message-2

…

The format of each DigiSAFE message is:

SOURCE = *source-DigiSAFE-name* -or- DESTINATION = *dest-DigiSAFE-name*

1 = DigiSAFE-bit-1

2 = DigiSAFE-bit-2

…

- *source-DigiSAFE-name* is the name of the transmitting DigiSAFE node. If this name matches one of the names in a DIGISAFE ID record in the VNT file, that node is identified as the transmitting node.
- *dest- DigiSAFE -name* is the name of the receiving DigiSAFE node. If this name matches one of the names in a DIGISAFE ID record in the VNT file, that node is identified as the receiving node. The messages in this section typically have either a source or a destination name.
- *DigiSAFE -bit-1* through *DigiSAFE -bit-N* are the message bits. Message bit numbers are set at 160 for DigiSAFE messages; the maximum bit number in the message must match the message length for this link in the Vital serial link definition file. Bit names are the variables associated with the bit. PERMZERO is allowed to indicate that a transmitted bit's value is always False.

NOTICE

DigiSAFE communications is only available in certain CAAs.

9.4.3 GTP Communications Section

This section consists of a section header followed by VSP-to-GTP and GTP-to-VSP messages. Each GTP board in the system typically has one VSP-to-GTP and one GTP-to-VSP message.

The data format is:

GTP COMMUNICATIONS SECTION

vsp-to-gtp1-message

gtp1-to-vsp-message

vsp-to-gtp2-message

gtp2-to-vsp-message

…

The format of a VSP-to-GTP message is:

GTP LINK *link-num* = **BLOCK***(block-num)*

DESTINATION = *gtp-name*

TRACK A

1 = trackA-bit-1

2 = trackA-bit-2

…

TRACK B

1 = trackB-bit-1

2 = trackB-bit-2

...

- *link-num* is a number from 1 to maximum, unique for all GTP boards and VSOE links in the system. The maximum link number is 200.
- *block-num* is a codeword block number from 0.01 to 17.00 defining a set of codewords unique for all GTP boards and VSOE links in the system. See the section on hardware application rules for information on link and block assignment.
- *gtp-name* is the name of the transmitting GTP board, matching the GTP ID in the .VPC file.
- *track bits* are the message bits. Message bit numbers can range from 1 to the maximum allowed for a given track on the GTP board. Bit names are the variables associated with the bits. PERMZERO is allowed to indicate that a bit's value is always False, or as a place-holder for a received bit.

The format of a GTP-to-VSP message is similar to the above but the DESTINATION record is replaced by a SOURCE record, indicating that the GTP board is sending the message to VSP.

9.4.4 CRG Communications Section

This section consists of a section header followed by a data message and a status message for each CRG board in the system. The data message is sent from the VSP board to the CRG board as a command for a specific code rate to be generated by the CRG output; the status message is returned to the VSP board from the CRG board as a non-vital indication that is True if the VSP output command was granted or False if the command was not granted.

The data format is:

CRG COMMUNICATIONS SECTION

crg-data-message-1

crg-status-message-1

crg-data-message-2

crg-status-message-2

…

WARNING

CRG STATUS PARAMETER APPLICATION

The CRG status parameters are calculated in a non-vital manner and must not be applied as fail-safe parameters.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

The CRG data message format is:

DESTINATION = *crg-name*

PORT 1

1 = port1-bit-1

2 = port1-bit-2

…

PORT 2

1 = port2-bit-1

2 = port2-bit-2

...

- *crg-name* is the name of the CRG board. If this name matches one of the names in a CRG ID record in the VPC file, that board is identified as the board receiving the output command from the VSP board (DESTINATION in this case refers to the fact that data is sent from the VSP board to the CRG board for output).
- *PORT* numbers can range from 1 to 8, and must be ordered sequentially.
- *port bits* are the message bits. Message bit numbers can range from 1 to the maximum allowed for a given port. Bit names are the variables associated with the bits. PERMZERO is allowed to indicate that a bit's value is always False.

The CRG status message format is:

SOURCE = *crg-name*

PORT 1 = *port1-bit*

PORT 2 = *port2-bit*

...

- *crg-name* is the name of the CRG board. If this name matches one of the names in a CRG ID record in the VPC file, that board is identified as the board transmitting the status back to the VSP board (SOURCE in this case refers to the fact that status is sent from the CRG board back to the VSP board).
- *PORT* numbers can range from 1 to 8, and must be ordered sequentially.
- *port bits* are the input variable names associated with each port's status bit. PERMZERO is allowed as a place-holder for the bit.

9.4.5 Example

```
NETWORK VSC SECTION
* outgoing VSOE-to-VSOE message (send-only, no receive message in this file)
SOURCE = BA4-BA3
1 = 12RGK-LA
2 = VRDFRNT-DI
3 = PERMZERO

DIGISAFE SECTION
* outgoing DigiSAFE message
SOURCE = BA4-ZC1
1 = 4NWZ
2 = 4RWZ
3 = PERMZERO
4 = PERMZERO
.
.
.
160 = PERMZERO

GTP COMMUNICATIONS SECTION
* VSP-to GTP message for code requests
GTP LINK 55 = BLOCK(3.0)
DESTINATION = THIS GTP BOARD
TRACK A
1 = P1-CODE-1
2 = P1-CODE-2

CRG COMMUNICATIONS SECTION
* VSP-to-CRG message for code requests
DESTINATION = THIS CRG BOARD
PORT 1
1 = P1-CODE-1
2 = P1-CODE-2
* CRG-to-VSP message for board status
SOURCE = THIS CRG BOARD
PORT 1 = P1-STATUS
```

9.5 VITAL LOGIC FILES (.PRM/.VTL)

These files contain the Vital logic parameter declarations and the application logic. In most cases, the parameter declarations can be moved into the logic file and the .PRM file is not needed. The CAAPE produces a .PRM file from graphics for consistency with the non-vital logic. The logic file should be the last one processed by the compiler.

The file format is:

current-result-parameter-declarations

self-latched-parameter-declarations

timer-parameter-declarations

timer-pgm-parameter-declarations

Boolean-equation-section

9.5.1 Current Result Parameter Declarations

This section is required if there are Current Result parameters in the logic. It consists of the section header CURRENT RESULT SECTION followed by a list of current result parameters on one or more lines. Parameter names can be separated by spaces or commas.

For example:

```
CURRENT RESULT SECTION
TEMP-PARM1, TEMP-PARM2
TEMP-PARM3
```

9.5.2 Self-Latched Parameter Declarations

This section is required if there are Self-Latched parameters in the logic. It consists of the section header SELF-LATCHED PARAMETER SECTION followed by a list of self-latched parameters on one or more lines. Parameter names can be separated by spaces or commas.

For example:

```
SELF-LATCHED PARAMETER SECTION
IVPI_PERM0 102_AR-LO 102_AR-LD 104R-F 104H2
IVPI_PERM1 102_AY-LO 102_AY-LD 106_AR-F 104H-OLD
```

9.5.3 Timer Parameter Declarations

This section is required if there are software Timers in the logic. It consists of the section header TIMER EXPRESSION RESULT SECTION for Permanently Programmed Vital Timers (PPVT) and TIMER PGM EXPRESSION RESULT SECTION for Field-Settable Software Vital Timers (FSSVT) followed by a list of timer parameters on one or more lines. Parameter names can be separated by spaces or commas.

For example:

```
TIMER EXPRESSION RESULT SECTION
102_AR-LDLY 104R-LDLY 106_AR-LDLY

TIMER PGM EXPRESSION RESULT SECTION
TMR1      TMR2  TMR3
```

9.5.4 Boolean Equation Section

This section contains the logic statements. It starts with the BOOLEAN EQUATION SECTION header record and can end with an optional END BOOLEAN EQUATION SECTION record. Available statement types are:

- Application Statement
- Location Statement
- Boolean Equation Statement
- Time Delay Statement
- Time Delay Programmable Statement

9.5.4.1 APPLICATION Statements

The statement format is:

APPLICATION = *application-info*

- *application-info* is any application information the user wishes to provide. If used, information on the application data record appears in the index report of the compiler listing with the number of the page on which this application record occurs.

For example:

APPLICATION = ROUTE LOCKING

APPLICATION = FLEETING

9.5.4.2 LOCATION Statements

The statement format is:

LOCATION = *location-info*

- *location-info* is any location information the user wishes to provide. If used, the information on the location data record appears in the index report of the compiler listing with the number of the page on which this location record occurs.

For example:

LOCATION = LOCATIONS 20,21,22

LOCATION = LOCATION 23

9.5.4.3 Boolean Equation Statements

The data for a single equation may be contained on several sequential lines. In addition, the equation may have a Vital time delay (see Time Delay Statements for additional information).

The statement format is:

BOOL *result-list* = (*logic-expression*)

SLOW *result-list* = (*logic-expression*)

- SLOW indicates a slow pick / slow drop relay function.
- *result-list* is a list of one or more result names. If there is more than one result name, they must be separated by commas. If the list of result names does not fit on a single data record, the last non-blank character on the record can be a comma and the list of result names can be continued on the next data record.
- *logic-expression* is the Boolean equation. It is surrounded by parentheses and separated from result names by an equals sign (=). The Boolean equation is composed of one or more parameter names separated by logical 'and' and 'or' operators. A logical 'and' is represented by an asterisk (*) and a logical 'or' is represented by a plus sign (+).

Equations need not be defined in sum-of-products form. Equations with nested parentheses or in product-of-sums form are expanded to sum-of-products form, and the compiler's Boolean Equation report shows both the original and expanded forms of the equation.

The complement of a parameter can be used by the addition of '.N.' in the front of the parameter to indicate a logical 'not'.

If the Boolean equation is to be continued on successive records, the last non-blank character on the continued record must be one of the operators (+ or *). Otherwise, the end of the equation is detected by the right parenthesis after the last parameter name.

For example:

```
BOOL 1ESTOP = (CONT13-NEG * 1ST + 1EGZ-NEG)
BOOL 1EAE-LDO, 2EAE-LDO = (.N.1-2EAEP)
BOOL 8LS = (8TP * (8L + .N.8RWC * .N.8NWC * 8LS))
SLOW 1L = (1ES * 1WS * 1TP)
```

9.5.4.4 Time Delay and Time Delay Programmable Statements

Time delay/Time Delay Programmable statements specify a Vital time delay function for a subsequent Boolean equation in the Vital Logic file. These data records must identify what kind of time delay to apply to the immediately following equation. An equation may be delayed by a fixed time delay.

The delay put into Time Delay Programmable statements contain the maximum value for that timer. The delay time in these statements can be modified using the AlsDload utility in the field and having the correct password.

There is an additional time delay option that is useful only when equivalent relay circuits are needed as an output. This time delay function is recognized only in conjunction with Vital logic, since it relies on a Vital one-second VSP cycle time. A slow release delay is identified by means of time delay records, which combine multiple equations onto a single circuit page with the slow release time delay specified. These slow release time records have no effect on the operation of VSP and no equations are generated as a result of these records; they are used only for information pertinent to a relay circuit's file. The records are not accepted unless a RELAY CIRCUITS record is present in the input file.

The statement format is:

TIME DELAY = *num-minutes* **MINUTES** , *num-seconds* **SECONDS**

TIME DELAY PROGRAMMABLE = *num-minutes* **MINUTES** , *num-seconds* **SECONDS**

TIME SLOW RELEASE START
TIME SLOW RELEASE END = num-seconds **SECONDS**

- *num-minutes* and *num-seconds*, for the first time delay format, (a fixed Vital time delay), is the number of minutes and seconds that must elapse before allowing the equation to evaluate as true. The equation parameters for at least one of the product terms of the equation must remain true for the specified period of time before the result parameter takes on a true value. The time limit may be set up to 59 minutes and 59 seconds. In addition, both minutes and seconds are not required, but at least one of them must be present.
- The second delay format shown is a record used to identify the initial equation in a group of equations which performs a slow release function. The result parameter(s) of this equation are saved to be the result(s) of the single slow release equation output to a relay circuit's file.
- *num-seconds* in the last delay format shown is the quantity of seconds of delay in the slow release function. The maximum slow release delay for this format is 59 seconds. This record identifies the final equation in a group of equations which performs a slow release function. The equation to the right of the equals (=) is saved to be the equation part of the single slow release equation output to a relay circuits file.

For example:

```
TIME DELAY = 3 MINUTES, 30 SECONDS
BOOL OE-TM = (IVPI_XFR*.N.OE*VRDFRNT-DI)

TIME DELAY PROGRAMMABLE = 1 MINUTES, 45 SECONDS
BOOL TMR1 = (VRDFRNT-DI)

TIME SLOW RELEASE START
BOOL MCH-SR = (MC-NVI + MCH-SR1)
BOOL MCH-SR1 = (MC-NVI + MCH-SR2)
BOOL MCH-SR2 = (MC-NVI + MCH-SR3)
BOOL MCH-SR3 = (MC-NVI + MCH-SR4)
TIME SLOW RELEASE END = 5 SECONDS
BOOL MCH-SR4 = (MC-NVI)
```

In the last example, the time delay records for the slow release function generate a single equation of the form "MCH-SR = MC-NVI" with a slow release time delay of 5 seconds. Any intervening equations are discarded, whether they are part of the slow release function or not. Therefore, it is important to group the pertinent slow release equations in a block with no intervening unrelated equations.

9.5.4.5 LIBRARY FILE Records

One or more of these records can be placed anywhere in the logic to indicate the path/ names of library files to access.

The record format is:

LIBRARY FILE = *file*

- *file* is the library file name or path.

For example:

```
LIBRARY FILE = D:\LIBFILES\CTA\NORTH.LIB
LIBRARY FILE = LIBFILE
```

9.5.4.6 LIBR Records

Use a LIBR statement to insert the contents of a library member into the logic and substitute application names for its generic names.

The record format is:

LIBR *[filename :] membername [(subst-list)]*

- *filename* is the optional library file name.
- *membername* is the library member name.
- *subst-list* is a list of actual names to be substituted for the generic ones defined for the library member. Multiple input lines are allowed, as long as each preceding line ends with a comma. All generic names in a library member must be given a corresponding actual name; names are assigned in the same order as the member's list of arguments. For example, in LIBR X(A,B,C), A is assigned to the first argument in the argument list, B is assigned to the second and C to the third.

For example:

```
LIBR SWITCH MEMBER 1
LIBR XLIBFILE : MEMBER A
LIBR XLIBFILE : SIGNAL STORAGE( 1WZ, 1XY )
```

SECTION 10 – COMM COMPILER FILES

This section discusses the files used specifically by the iVPI Comm Compiler. The VSOE Node Declaration (.VNT) and the Link Definition (.CW) files are shared with the iVPI Compiler and are described in that section.

10.1 MAIN COMPILER FILE (.VCC)

This file contains documentation records for the Vital Comm application and INCLUDE records referencing additional files with the rest of the Vital Comm application data. The INCLUDE records usually follow any documentation records in the file.

10.1.1 Revision History

Application revision history can be stored as special comments in the main input file.

The format is:

```
*%REV id : date : author : summary
*+ details
```

- *id* is a unique version identifier.
- *date* is the revision date.
- *author* is the revision author.
- *summary* is a short description of the change.
- *details* are optional records detailing the change. Details records must follow the revision header record, but other comments can be placed between revisions.

For example:

```
**********************************************
*%REV A : 03/04/07 : JRM : INITIAL
**********************************************
*%REV B : 05/04/07 : JRM : RELEASE
*+ ADD SECOND VSOE INTERFACE
**********************************************
```

10.1.2 File Records

10.1.2.1 APPLICATION PROGRAM NUMBER Record

This record provides the top-level archive name or part number for the entire iVPI system. In addition to the name or number, the current revision letter of the file and the initials of the person responsible for the latest updates can be included on this record. This record is optional.

The record format is:

APPLICATION PROGRAM NUMBER = *program-num*, REV.*rev*

- *program-num* is an archive name of up to 13 characters or an Alstom drawing number.
- *rev* is whatever revision information is needed to document the system. For instance, the current revision date, revision letter, and the initials of the person responsible for making the latest updates could be included in this field. A maximum of 21 characters are saved for this field, which is output in the headings of all documentation generated by the compiler. The main text string from (data-1) is placed on the module I/O labels for VSP and NVSP boards.

For example:

APPLICATION PROGRAM NUMBER = 32917-001 GR 00, REV. C, JKL

APPLICATION PROGRAM NUMBER = ARCHIVENAME, REV. D, JRM

10.1.2.2 VSPCP PROGRAM NUMBER Record

This record provides the archive name or part number of this VSP Comm application program. This means the number assigned to the specific VSP Comm application code, not the system software drawing number (see the SYSTEM SOFTWARE Record description). In addition to the name or number, the current revision letter of the file and the initials of the person responsible for the latest updates should be included on this record.

The record format is:

VSPCP PROGRAM NUMBER = *program-num*, REV.*rev*

- *program-num* is an archive name of up to 13 characters or an Alstom drawing number assigned to the program.
- *rev* is whatever revision information is needed to document the system. For instance the current revision date, revision letter, and the initials of the person responsible for making the latest updates could be included in this field. A maximum of 21 characters are saved for this field, which is output in the headings of all documentation generated by the compiler.

For example:

VSPCP PROGRAM NUMBER = 32917-001 GR 00, REV. C, JKL

VSPCP PROGRAM NUMBER = IVPIPROGNAME, REV. D, JRM

10.1.2.3 COPYRIGHT YEAR Records

This record denotes the year of copyrighting of the VSPCP Program. This means the copyright of the application code (VSPCP PROGRAM NUMBER), not the system software copyright date.

The record format is:

COPYRIGHT YEAR = *year*

- *year* is the 4-digit date of software copyright.

For example:

COPYRIGHT YEAR = 2007

10.1.2.4 SYSTEM SOFTWARE Records

This record contains the Alstom drawing number of the VSP Comm system software, the routines that run using the network communications application data generated by the Compiler Program. This is not the drawing number assigned to the specific application code (see Section 9.1.2.3 VSPCP PROGRAM NUMBER Records).

This record is required for the VSP Comm Compiler to generate the application data / system software EPROM file. The system software part number is used to determine which comm system software to use when generating the file. If an invalid part number is entered, the Compiler is unable to find the system software and therefore unable to create the EPROM file.

The record format is:

SYSTEM SOFTWARE = *partnum*

- *partnum* is the Alstom drawing number identifying the system software. A revision letter is required.

For example:

SYSTEM SOFTWARE = 40025-401 GR 00, REV. A

10.1.2.5 CONTRACT NUMBER Records

This record identifies the contract number(s) for which this VSP Comm Program is being provided.

The record format is:

CONTRACT NUMBER = *contract-num*

- *contract-num* is the contract number(s), maximum of 40 characters in length.

For example:

CONTRACT NUMBER = 91-79897

CONTRACT NUMBER = 91-79400,91-80500

CONTRACT NUMBER = PD221

10.1.2.6 CONTRACT NAME Records

This record identifies the contract name(s) for which this VSP Comm Program is being provided.

The record format is:

CONTRACT NAME = *contract-name*

- *contract-name* is the contract name(s), a maximum of 40 characters in length.

For example:

CONTRACT NAME = KENTON AVE. CHICAGO

10.1.2.7 CUSTOMER NAME Records

This record identifies the customer name for whom this VSP Comm Program is provided.

The record format is:

CUSTOMER NAME = *cust-name*

- *cust-name* is the customer's name, a maximum of 40 characters in length.

For example:

CUSTOMER NAME = CONRAIL

10.1.2.8 EQUIPMENT LOCATION Records

This record identifies the physical location of the equipment for which this VSP Comm Program is being provided.

The record format is:

EQUIPMENT LOCATION = *location*

- *location* is the physical location of the iVPI module(s) in which this program is located, a maximum of 40 characters in length.

For example:

EQUIPMENT LOCATION = OLD RELAY ROOM, RACK A3

10.1.2.9 DESIGNER Records

This record identifies the name(s) of the individual(s) responsible for the design of this VSP Comm Program.

The record format is:

DESIGNER = *name*

- *name* is the name(s) of the designer(s), a maximum of 40 characters in length.

For example:

```
DESIGNER = JOHN Q. DESIGNER
```

10.1.2.10 CHECKER Records

This record identifies the name(s) of the person(s) responsible for checking all aspects of the iVPI Comm compiler input file.

The record format is:

CHECKER = *name*

- *name* is the name(s) of the individual(s) responsible for checking these equations, a maximum of 40 characters in length.

For example:

```
CHECKER = JUNE D. CHECKER
```

10.1.2.11 VSP ID Records

This optional record provides a user-readable name for the VSP board; the name can be used to identify the board on a network.

The record format is:

VSP ID = *board-name*

- *board-name* is a board name of up to 40 characters.

For example:

```
VSP ID = IVMAL1A
```

10.1.2.12 ENET1, ENET2 Device Records

These records specify the characteristics of the network devices on the VSP board.

The record format is:

ENET1 DEVICE = IP*(ip-address)* [,**MASK***(subnet-mask)*,

DIAGNOSTICS*(diags)*]

ENET2 DEVICE = **IP***(ip-address)* [,**MASK***(subnet-mask)*,

DIAGNOSTICS*(diags)*]

- ENET1 and ENET2 identify the network devices. The appropriate record(s) must exist depending on which devices are to be used; if one of the records does not exist, the corresponding device is assumed to be unused. At least one network device must be used in the application.
- *ip-address* is the IP address assigned to the device, up to 40 characters in length. This field is required if the record exists.
- *subnet-mask* is the mask which, together with the IP address, determines the subnet on which this device resides. It is available only for CAA versions with subnets/ redundancy. This field is optional; if it does not exist, the default mask of 25.255.255.0 will be assigned.
- *diags* is YES or NO, indicating whether the device is used for network-based diagnostics. This field is optional; if it does not exist, a default of no diagnostics is assigned. The MACTCP record should be used instead for CAA versions with subnets/redundancy.

For example:

ENET1 DEVICE = IP(162.21.17.19),DIAGNOSTICS(YES)

ENET2 DEVICE = IP(162.21.17.20), MASK(255.255.255.0)

10.1.2.13 MACTCP Records

This record is used by CAA versions with subnets/redundancy to specify the characteristics of the MACTCP interface to MMS. The record format is

MACTCP = **DEVICE**(*device*), **MMS***(mms-name)* [,**PORT**(*socket-port*),

GATEWAY*(gateway-name*)]

- *device* is the network device that will be used: ENET1, ENET2 or REDUNDANT for path-redundant MMS links using both network devices.
- *mms-name* is the user name of the remote MMS. A corresponding MMS record must be found in the MMS CONNECTION NAME SECTION.
- *socket-port* is MACTCP-N for automatic assignment according to the new Ethernet ports scheme, MACTCP for automatic assignment according to the old scheme, or one or two numeric entries depending on whether the link is redundant.
- *gateway-name* is the user name(s) of the gateways to be used if the remote MMS is on a different subnet. A corresponding GATEWAY record must be found in the GATEWAYS SECTION. Two names are required if the link is redundant; if only one path of a redundant link requires a gateway, the name for the other path should be NONE.

For example:

```
MACTCP = DEVICE(ENET1),MMS(MMS-1), PORT(MACTCP-N)
MACTCP = DEVICE(REDUNDANT),MMS(MMS-R), PORT(1600,1601),
         GATEWAY(GW-1,NONE)
```

10.1.2.14 INCLUDE Record

These records are used to identify additional files for the Compiler to read. INCLUDE records are generally the last non-comment records in the file.

The record format is:

INCLUDE = *file*

- *file* is the path/name of the file to be read. If there is no directory information, the Compiler looks for the file in the same directory as the main file. If the file name has embedded spaces, it must be enclosed in quotes.

For example:

```
INCLUDE NETWORK.VNT
INCLUDE NETWORK.CW
INCLUDE NETWORK.NVS
```

10.2 VSOE AND DIGISAFE CONNECTION DATA FILE (.NVS)

This file specifies the network properties of the VSOE, and DigiSAFE nodes in the network. The nodes used by a given system are identified by matching their names against the names in the .VNT and .CW files:

- If a VSOE or DigiSAFE name matches one of the names in the .VNT file, the node is a local one used by the system being compiled. Its network properties are saved and the node is assigned to the ENET1 or ENET2 network device depending on the IP address.
- If a VSOE (or DigiSAFE) name does not match a local VSOE (or DigiSAFE) node but is identified in the .CW file as a remote source or destination of a link involving a local VSOE (or DigiSAFE), the node is a remote one. Its network properties are stored to allow the local node to be able to establish a link with it.

The file consists of a section header followed by a series of VSOE, ATCV and DigiSAFE properties declarations. The file format is

VSOE CONNECTION DATA SECTION

node-properties-1

node-properties-2

…

DigiSAFE-node-properties-3

DigiSAFE-node-properties-4

…

The format of a property record for a VSOE node is:

VSOE = **NAME***(vsoe-name)*, [**REDUNDANT**,] **IP***(vsoe-ip)*, **PORT***(vsoe-port)* [, **MASK***(subnet-mask)*]

- *vsoe-name* is the name of the VSOE node. It must match one of the names in the .VNT file for the record to be used. **REDUNDANT** indicates that links to this node are redundant. It is available only for CAA versions with subnets/redundancy.
- *Vsoe-ip* specifies the IP address(es) of the node. If redundant, two IP addresses separated by a comma are required. If the node is a local one, i.e. if it exists on this system, the IP addresses must match the IP addresses of the network devices which handle the communications for this node. If the node is a remote one, the IP addresses will be used to set up communications with it.
- *Vsoe-port* identifies the Ethernet socket port used by the node. Available entries are
  - ENET1-N, ENET2-N, REDUNDANT-N: identifies the network device(s) used by the local or remote node. The CAA automatically assigns port values using the new ports assignment scheme (50000 range). These entries are only available for CAA versions 610 and later with subnets/redundancy. When available, they are the preferred values unless this node is to communicate with a node in an existing application compiled using a CAA version before 610.
  - ENET1, ENET2: identifies the network device used by the local or remote node. The CAA automatically assigns values using the old ports assignment scheme (1000 range). These entries must be used for CAA versions before 610 with no subnets/redundancy, or can be used for CAA versions610 and later if the old port values must be retained for communicating with a VSOE node in an existing application.
  - One or two numeric values depending on whether the node is redundant.
- *subnet mask* designates the mask(s) which when combined with the IP address data, specifies the subnet(s) on which this node resides. It is available only for CAA versions with subnets/redundancy. If redundant, two masks separated by a comma are required. This field is optional; if it does not exist, default values of 255.255.255.0 will be assigned.

The fields in this record can be placed on multiple lines if each line but the last ends with a comma.

The format of a property record for a DigiSAFE node is:

DIGISAFE = **NAME***(DigiSAFE-name)*, [**REDUNDANT**,] **IP***(DigiSAFE-ip)*,

PORT*(DigiSAFE-port)* [, **MASK***(subnet-mask)*]

- *DigiSAFE-name* is the name of the DigiSAFE node. It must match one of the names in the .VNT file for the record to be used.
- **REDUNDANT** indicates that links to this node are redundant. It is available only for CAA versions with subnets/redundancy. It must be defined for DigiSAFE nodes.
- *DigiSAFE-ip* specifies the IP address(es) of the node. If redundant, two IP addresses separated by a comma are required. If the node is a local one, i.e. if it exists on this system, the IP addresses must match the IP addresses of the network devices which handle the communications for this node. If the node is a remote one, the IP addresses will be used to set up communications with it.
- *DigiSAFE-port* identifies the network port used by the node. A value of REDUNDANT-N: identifies the network device(s) used by the local or remote node. The CAA automatically assigns port values of 61440 for default DigiSAFE ports. Numeric values can also be assigned if the DigiSAFE ports differ from the default values. *subnet mask* designates the mask(s) which, when combined with the IP address data, specifies the subnet(s) on which this node resides. It is available only for CAA versions with subnets/redundancy. If redundant, two masks separated by a comma are required. This field is optional; if it does not exist, default values of 255.255.255.0 will be assigned.

The fields in this record can be placed on multiple lines if each line but the last ends with a comma.

For example:

```
VSOE CONNECTION DATA SECTION

VSOE = NAME(IVMA-OS1),IP(172.16.21.17),PORT(ENET1-N)
VSOE = NAME(OS1-IVM),IP(172.16.29.10),PORT(ENET1)
VSOE = NAME(VSOE-1-RED),REDUNDANT,IP(172.16.29.10,173.16.29.10),
PORT(REDUNDANT-N),MASK(255.255.254.0,255.255.254.0)


DIGISAFE = NAME(DS_01),REDUNDANT,IP(120.80.55.1,120.80.55.2),
 PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(REM_ZC_01A),REDUNDANT,IP(120.80.55.6,120.80.55.10),
 PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
DIGISAFE = NAME(REM_ZC_01B),REDUNDANT,IP(120.80.55.7,120.80.55.9),
 PORT(REDUNDANT-N),MASK(255.255.255.0,255.255.255.0)
```

NOTICE

DigiSAFE communications is only available in certain CAAs.

10.3 GATEWAYS FILE (.GW)

This file specifies the network properties of the gateways in the network. It is only available for CAA versions with subnets/redundancy. The gateways used by a given system are identified by matching their names against the gateway names in the various records that map network connections across subnets.

The file format is

GATEWAYS SECTION

gateway-1

gateway-2

…

The format of a property record for a gateway is:

GATEWAY = NAME*(gateway-name)*, [IP*(gateway-ip)*]

- *gateway-name* is the name of the gateway. It must match the gateway name in a network connection record to be used. Maximum size is 20 characters.
- *gateway-ip* specifies the IP address of the gateway. This field is optional; if it does not exist, a direct connection is assumed and the local and remote devices must be on the same subnet.

The fields in these records can be placed on multiple lines if each line but the last ends with a comma.

For example:

GATEWAYS SECTION

GATEWAY = NAME(GW-1),IP(172.16.21.17)

GATEWAY = NAME(NULL-GW)

10.4 MMS CONNECTION DATA FILE (.NMM)

This file specifies the network properties of the MMS in the network. It is only available for CAA versions with subnets/redundancy. The MMS used by a given system are identified by matching their names against the MMS names in the MACTCP records in .VCC and .CSI files.

The file format is

MMS CONNECTION DATA SECTION

mms-properties-1

mms-properties-2

…

The format of a property record for an MMS is:

MMS = **NAME**(*mms-name*), [**REDUNDANT**,] **IP**(*mms-ip*) [,

MASK(subnet-mask)]

- *mms-name* is the name of the MMS. It must match the MMS name in a MACTCP record to be used.
- *REDUNDANT* indicates that links to this MMS are redundant.
- *mms-ip* specifies the IP address(es) of the MMS. If redundant, two IP addresses separated by a comma are required.
- *subnet mask* designates the mask(s) which when combined with the IP address data, specifies the subnet(s) on which this MMS resides. If redundant, two masks separated by a comma are required. This field is optional; if it does not exist, default values of 255.255.255.0 will be assigned.

The fields in these records can be placed on multiple lines if each line but the last ends with a comma.

For example:

```
MMS CONNECTION DATA SECTION


MMS = NAME(MMS-1),IP(172.16.21.17)
MMS = NAME(MMS-2),REDUNDANT,IP(172.16.21.18,173.17.22.18),
MASK(255.255.255.0,255.255.254.0)
```

SECTION 11 – NVSP COMPILER FILES

This section discusses the files used specifically by the NVSP Compiler. The Hardware and VSP-NVSP Communications files are shared with the iVPI Compiler and are described elsewhere.

11.1 MAIN COMPILER FILE (.CSI)

This file contains documentation records for the non-vital application and INCLUDE records referencing additional files with the rest of the non-vital application data. The INCLUDE records usually follow any documentation records in the file.

11.1.1 Revision History

Application revision history can be stored as special comments in the main (.VPC or .CSI) compiler input file. These comments are read by the Relay Equivalent Drawing Program.

The data format is:

```
*%REV id : date : author : summary
*+ details
```

- *id* is a unique version identifier.
- *date* is the revision date.
- *author* is the revision author.
- *summary* is a short description of the change.
- *details* are optional records detailing the change. Details records must follow the revision header record, but other comments can be placed between revisions.

For example:

```
**********************************************
*%REV A : 03/04/05 : JRM : INITIAL
**********************************************
*%REV B : 05/04/05 : JRM : RELEASE
*+ ADD SECOND VSC INTERFACE
**********************************************
```

11.1.2 File Records

11.1.2.1 COMPILER RUN CONTROLS Records

This record comprises a list of override commands to control the processing of the NVSP compiler program. If used, this record must be the first non-comment data record in the compiler input file. This record is ordinarily not needed, since run controls can now be specified through CAAPE's user interface.

The record format is:

COMPILER RUN CONTROLS = *data-1, data-2, data-n*

- *data-1* though *data-n* are a list of compiler run controls chosen from Table 10–1. If the list exceeds the limit of one data record, it may be continued in subsequent data records provided the last non-blank character on the preceding data record is a comma (,). The order of the output reports is predetermined by the compiler. These run controls only indicate whether or not the report is to be generated, and do not define order. Commands are processed in the order entered on the data record. It is possible to both select a given command and then accidentally cancel it if the command is specified in both its positive and negative sense.

For example:

```
COMPILER RUN CONTROLS = LIST ALL,NO XREF
COMPILER RUN CONTROLS = NO LIST ALL,PROM
```

Table 11–1 lists the run control commands along with their definitions.

NOTICE

When performing an initial build, the last-installed CAA version is automatically used to do the compile. This is generally the version that uses the latest available Vital system software. If a different CAA version is desired, or to use non-default compiler options, do a Make Files first to create the applications and then go to the ***FileView*** and set the run controls for each application that was created. See P2512A CAAPE User Manual for more details.

Table 11–1. NVSP Compiler Run Control Commands

Command	Action
LIST ALL NO LIST ALL	Permits the user to enable or disable all of the Compiler output reports. This command may be specified as the general condition for all output reports, and then overridden for specific reports by supplying the proper run control. The NO LIST ALL run control is the default option for the Compiler. These controls are not affected by this command and must be uniquely specified to be enabled: PROM, GRAPHISM.
BOOLEAN NO BOOLEAN	Controls the Boolean equation section report. This report lists the user's Boolean equation section data records.
PARAM NO PARAM	Controls the listing of the parameter name report. This report lists parameter and I/O names and their internal address assignments.
XREF NO XREF	Controls the listing of the parameter name and I/O cross reference report. This report provides a listing in alphanumeric character order of every parameter, input and output name used in the compilation, and its type. In addition, the report shows the use of the names in the expressions.
INDEX NO INDEX	Controls the listing of the index report. This report provides an index to the Compiler listing by page number, giving the starting page number of each report.
PROM	Controls the generation of an output file of NVSP PROM code. No listing report is produced. This run control must be uniquely specified, it is not enabled by the LIST ALL run control.
GRAPHSIM	Causes a Graphical Simulator data file to be created. This run control must be uniquely specified, it is not enabled by LIST ALL.
NOERRCODE	On NVSP this run control disables the generation of PROM code for array bounds checking. Array bounds checking prevents non-vital application logic from trying to access memory outside the bounds of an array if an improper array index is used. The code to do this takes up PROM space and takes time to execute, but may help prevent unpredictable behavior if a bad array index is accidentally used in the logic. This run control must be uniquely specified, it is not enabled by LIST ALL.

11.1.2.2 APPLICATION PROGRAM NUMBER Records

This record provides the top-level archive name or part number for the entire iVPI system. In addition to the name or number, the current revision letter of the file and the initials of the person responsible for the latest updates can be included on this record. This record is optional.

The record format is:

APPLICATION PROGRAM NUMBER = *program-num*, REV.*rev*

- *program-num* is an archive name of up to 13 characters or an Alstom drawing number.
- *rev* is whatever revision information is needed to document the system. For instance the current revision date, revision letter, and the initials of the person responsible for making the latest updates could be included in this field. A maximum of 21 characters are saved for this field, which is output in the headings of all documentation generated by the compiler. The main text string from (data-1) is placed on the module I/O labels for VSP and NVSP boards.

For example:

APPLICATION PROGRAM NUMBER = 32917-001 GR 00, REV. C, JKL

APPLICATION PROGRAM NUMBER = ARCHIVENAME, REV. D, JRM

11.1.2.3 NVSP PROGRAM NUMBER Records

This record provides the archive name or Alstom drawing number of the application logic for each NVSP board. This means the drawing number assigned to the specific application code, NOT the system software drawing number. In addition to the name or drawing number, the current revision letter of the file and the initials of the person responsible for the latest updates can be included on this record.

The record format is:

NVSP NVSP-*id* PROGRAM NUMBER = *program-num* , REV.*rev*

- *NVSP-id* is the number assigned to the NVSP board on the slot assignment record which must be 1, 2, 3, or 4.
- *program-num* is an archive name of up to 13 characters or an Alstom drawing number assigned to the program.
- *rev* is whatever revision information is needed to document the system. For instance the current revision date, revision letter and the initials of the person responsible for making the latest updates could be included in this field.

For example:

NVSP 1 PROGRAM NUMBER = 32917-037 GR 00, REV. A, MNO

NVSP 1 PROGRAM NUMBER = NVSPNAME, REV. A, MNO

11.1.2.4 COPYRIGHT YEAR Record

This record denotes the year of copyrighting of the NVSP Program. This means the copyright of the application code (NVSP PROGRAM NUMBER), NOT the system software copyright date.

The record format is:

COPYRIGHT YEAR = *year*

- *year* is the 4-digit date of software copyright.

For example:

COPYRIGHT YEAR = 1985

11.1.2.5 CONTRACT NUMBER Records

This record identifies the contract number(s) for which this NVSP Program is being provided.

The record format is:

CONTRACT NUMBER = *contract-num*

- *contract-num* is the contract number(s), maximum of 40 characters in length.

For example:

CONTRACT NUMBER = 91-79897

CONTRACT NUMBER = 91-79400,91-80500

CONTRACT NUMBER = PD221

11.1.2.6 CONTRACT NAME Record

This record identifies the contract name(s) for which this NVSP Program is being provided.

The record format is:

CONTRACT NAME = *contract-name*

- *contract*-name is the contract name(s), a maximum of 40 characters in length.

For example:

CONTRACT NAME = KENTON AVE. CHICAGO

11.1.2.7 CUSTOMER NAME Records

This record identifies the customer name for whom this NVSP Program is being provided.

The record format is:

CUSTOMER NAME = *cust-name*

- *cust-name* is the customer's name, a maximum of 40 characters in length.

For example:

CUSTOMER NAME = CONRAIL

11.1.2.8 EQUIPMENT LOCATION Records

This record identifies the physical location of the equipment for which this NVSP Program is being provided.

The record format is:

EQUIPMENT LOCATION = *location*

- *location* is the physical location of the iVPI module(s) in which this program is located, a maximum of 40 characters in length.

For example:

EQUIPMENT LOCATION = OLD RELAY ROOM, RACK A3

11.1.2.9 DESIGNER Record

This record identifies the name(s) of the individual(s) responsible for the design of this NVSP Program.

The record format is:

DESIGNER = *name*

- *name* is the name(s) of the designer(s), a maximum of 40 characters in length.

For example:

DESIGNER = JOHN Q. DESIGNER

11.1.2.10 CHECKER Records

This record identifies the name(s) of the person(s) responsible for checking all aspects of the NVSP compiler input file, including the application logic, the I/O, and the parameter definitions.

The record format is:

CHECKER = *name*

- *name* is the name(s) of the individual(s) responsible for checking these equations, a maximum of 40 characters in length.

For example:

CHECKER = JUNE D. CHECKER

11.1.2.11 NVSP ID Record

These records assign unique names to NVSP boards so communication messages can be assigned to a source NVSP and/or a destination NVSP. One of these records is required as input for each NVSP board in a system.

The record format is:

NVSP NVSP-*id* ID = NVSP-*name*

- *NVSP-id* is the number assigned to the NVSP board on the hardware slot assignment record which must be 1, 2, 3, or 4.
- *NVSP-name* is a 40-character name given to the board. This name should describe the function of the board or the board's location, if possible. The names for each NVSP board at an installation must be unique even if the NVSP boards are in different modules connected serially.

For example:

NVSP 1 ID = LOCAL CONTROL PANEL DRIVER

NVSP 2 ID = MAIN CODE UNIT

NVSP 3 ID = STANDBY CODE UNIT

11.1.2.12 NVSP SYSTEM SOFTWARE Records

This record contains the Alstom drawing number of the system software for the NVSP board: the top-level routines which execute the application logic. This is NOT the drawing number assigned to the specific application code (see Section 9.1.2.18 NVSP PROGRAM NUMBER Records).

The record format is:

NVSP NVSP-*id* SYSTEM SOFTWARE = *partnum*

- *NVSP-id* is the number assigned to the NVSP board on the slot assignment record which must be 1, 2, 3, or 4.
- *partnum* is the Alstom drawing number assigned to the program.

For example:

NVSP 1 SYSTEM SOFTWARE = 40026-107 GR 00

NVSP 2 SYSTEM SOFTWARE = 40026-107 GR 00

11.1.2.13 DATA LOGGING Records

This optional record indicates whether the data logger is to be enabled or not. Data logging structures are created if this record is used, and data logging is enabled if the status is "on", and disabled if the status is "off." The record is required in the documentation section if the data logger is used.

The record format is:

DATA LOGGING = *status*

- *status* is either "on", if data logging is enabled, or "off" if disabled.

For example:

DATA LOGGING = ON

DATA LOGGING = OFF

11.1.2.14 DIAGNOSTIC PASSWORD RECORD Records

This optional record identifies a password for use in protecting certain diagnostic functions. Entering the password into the NVSP MAC port grants access to the protected diagnostics.

NOTICE

If there is no MAC port activity for more than 30 minutes, however, the system automatically disables the diagnostic password forcing authorized personnel to reenter it.

The record format is:

DIAGNOSTIC PASSWORD = *password*

- *password* is an alphanumeric entry from one to eight characters long. Embedded spaces are allowed.

For example:

DIAGNOSTIC PASSWORD = DIAG USR

11.1.2.15 DIAGNOSTIC TERMINAL TYPE Records

This record is used to identify the type of diagnostic terminal to be used. It is required before any other diagnostic terminal information can be entered.

The record format is:

DIAGNOSTIC TERMINAL TYPE = *type*

- *type* is the terminal type: either HHT for hand-held, or MAC for VT100 or equivalent CRT.

For example:

DIAGNOSTIC TERMINAL TYPE = MAC

11.1.2.16 DIAGNOSTIC TERMINAL BAUD RATE Records

This optional record is used to set the baud rate for communication with the diagnostic terminal. It must be preceded by a DIAGNOSTIC TERMINAL TYPE record when used. Default value is 9600 baud if this record is not entered.

NOTICE

In order for AlsDload to work properly the Diagnostic settings must be as follows:
Diagnostic Terminal Baud Rate = 19200
Diagnostic Terminal Data Bits = 8
Diagnostic Terminal Stop Bits = 1
Diagnostic Terminal Parity = NONE

The record format is:

DIAGNOSTIC TERMINAL BAUD RATE = *baud-rate*

- *baud-rate* is the baud rate. Valid values for all boards are 75,110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 and 57600.

For example:

DIAGNOSTIC TERMINAL BAUD RATE = 1200

11.1.2.17 DIAGNOSTIC TERMINAL DATA FORMAT Records

This optional record is used to set the data format for communication with the diagnostic terminal. It must be preceded by a DIAGNOSTIC TERMINAL TYPE record when used. Default values are 8 data bits, 1 stop bit and no parity if no record is entered.

The record format is:

DIAGNOSTIC TERMINAL DATA FORMAT = *data-bits, stop-bits, parity*

- *data-bits* is the number of data bits, 7 or 8.
- *stop-bits* is the number of stop bits, 1 or 2.
- *parity* is a letter indicating parity type: E for even, O for odd, N for none.

For example:

```
DIAGNOSTIC TERMINAL DATA FORMAT = 8,1,N
```

11.1.2.18 LIBRARY PATH Records

One of these optional records can be placed anywhere in the documentation section to indicate the path and/or library file name to be used when accessing library data in the non-vital logic. It should be used only when a single path is being searched or a single library file is used.

The record format is:

LIBRARY PATH = *path [\ filename]*

- *path* is the path to be searched.
- *filename* is an optional file name, which can be included if a single library file is to be accessed. If no extension is given, '.LIB' is assumed.

For example:

```
LIBRARY FILE = D;\LIBFILES\CTA\NORTH.LIB
LIBRARY FILE = LIBFILE
```

11.1.2.19 ENET1, ENET2 Device Records

These records specify the characteristics of the network devices on the NVSP board.

The record format is:

ENET1 DEVICE = IP*(ip-address)* [,**MASK***(subnet-mask)*,
DIAGNOSTICS*(diags)*]

- ENET1 and ENET2 identify the network devices. The appropriate record(s) must exist depending on which devices are to be used; if one of the records does not exist, the corresponding device is assumed to be unused. At least one network device must be used in the application.
- *ip-address* is the IP address assigned to the device, up to 40 characters in length. This field is required if the record exists.
  - subnet-mask is the mask which, together with the IP address, determines the subnet on which this device resides. It is available only for CAA versions with subnets/redundancy. This field is optional; if it does not exist, the default mask of 25.255.255.0 will be assigned.
- *diags* is YES or NO, indicating whether the device is used for network-based diagnostics. This field is optional; if it does not exist, a default of no diagnostics is assigned. The MACTCP record should be used instead for CAA versions with subnets/redundancy.

For example:

```
ENET1 DEVICE = IP(162.21.17.19),DIAGNOSTICS(YES)
ENET2 DEVICE = IP(162.21.17.20), MASK(255.255.255.0)
```

11.1.2.20 MACTCP Records

This record is used by CAA versions with subnets/redundancy to specify the characteristics of the MACTCP interface to MMS. The record format is

MACTCP = DEVICE(device), **MMS**(mms-name) [,**PORT**(socket-port), **GATEWAY**(gateway-name)]

- *device* is the network device that will be used: ENET1, ENET2 or REDUNDANT for path-redundant MMS links using both network devices.
- *mms-name* is the user name of the remote MMS. A corresponding MMS record must be found in the MMS CONNECTION NAME SECTION.
- *socket-port* is MACTCP-N for automatic assignment according to the new Ethernet ports scheme, MACTCP for automatic assignment according to the old scheme, or one or two numeric entries depending on whether the link is redundant.
- *gateway-name* is the user name(s) of the gateways to be used if the remote MMS is on a different subnet. A corresponding GATEWAY record must be found in the GATEWAYS SECTION. Two names are required if the link is redundant; if only one path of a redundant link requires a gateway, the name for the other path should be NONE.

For example:

MACTCP = DEVICE(ENET1),MMS(MMS-1), PORT(MACTCP-N)

MACTCP = DEVICE(REDUNDANT),MMS(MMS-R), PORT(1600,1601),

GATEWAY(GW-1,NONE)

11.1.2.21 VSP SOFTWARE SITE ID / VSP SYSTEM ID Records

One of these optional records can be used to associate the NVSP application with a site ID read from the VSP board. The NVSP board runs only if the site ID switches for the VSP board have been set to the expected value specified in this record. If this record is not present, site ID is not checked.

The record formats are:

VSP SOFTWARE SITE ID = *site-id*

or

VSP SYSTEM ID = *sys-id*

- *site-id* is a 4-digit decimal number, 01-1022.
- *sys-id* is a 5-digit decimal number from 1 to 65534, or a hexadecimal number from 0X0000 to 0XFFFE (the "0X" prefix indicates that the data is in hexadecimal format). This value gives the entire site and revision ID data of the VSP board. The NVSP board ignores the revision ID portion and substitute its own revision ID.

For example:

VSP SOFTWARE SITE ID = 1014

VSP SYSTEM ID = 256

VSP SYSTEM ID = 0X0100

11.1.2.22 SOFTWARE REVISION ID Records

NOTICE

This required record identifies the revision of the NVSP application. The NVSP board runs only if the revision ID switches for the board have been set to the expected value specified in this record.

The record format is:

SOFTWARE REVISION ID = *rev-id*

- *rev-id* is a 2-digit decimal number, 01-62, or a hexadecimal number from 0X01 to 0x3E (the "0X" prefix indicates that the data is in hexadecimal format).

For example:

SOFTWARE REVISION ID = 24

SOFTWARE REVISION ID = 0X18

11.1.2.23 INCLUDE Records

These records are used to identify additional files for the Compiler to read. INCLUDE records are generally the last non-comment records in the file.

The record format is:

INCLUDE = *file*

- *file* is the path/name of the file to be read. If there is no directory information, the Compiler looks for the file in the same directory as the main file. If the file name has embedded spaces, it must be enclosed in quotes.

For example:

INCLUDE MYAPP.HDW

INCLUDE “NVSP2 APP.VC1”

11.2 SERIAL COMMUNICATIONS FILE (.CSS)

This part of the manual covers the CAA inputs required to specify both code system emulation and serial communications protocols. These terms are highly related: a code system emulation is just a specialized serial protocol developed for communication between an office and one or more field units on a railroad. For this reason, the CAA does not distinguish between the two. When the CAA manual may be referring to either, the generic term "protocol" is used.

Several protocols are available through the CAA. No effort is made in this manual to list all available protocols, or to describe all the rules for their application. This section of the manual lists CAA inputs that are common to most or all protocols. Refer to the appropriate documents for information specific to the actual protocols being used.

The .CSS file consists of a section header followed by one or more port definitions. Either the NVSP CODE SYSTEM SECTION or the SERIAL COMMUNICATIONS SECTION header can be used to start the section. These headers are interchangeable.

The file format is:

SERIAL COMMUNICATIONS SECTION

serial-port-definition-1

serial-port-definition-2

…

11.2.1 Serial Port Definition Section

A serial port definition section must exist for each port to be used in the system. Each port definition has a port header followed by port settings records and the input, output and special messages for the port. The port is not processed unless it has at least one useable input or output message.

The format for each serial port definition is:

SERIAL PORT *port* = **TYPE** *(protocol-name) [, port-options]*

port-settings

message-1

message-2

…

message-n

- *port* is the serial port to receive and transmit messages and is a decimal number from 1 to 5.
- *protocol-name* names the type of code system to be emulated on the port, or protocol for serial communications with another device. See the appropriate protocol manual for the name needed to indicate the desired protocol.
- *port-options* refers to any options specific to a given protocol. Two options are available to all protocols: **UNLATCHED CONTROLS** (application clears all controls for this port once per application cycle) and **LATCHED CONTROLS** (controls are not automatically cleared). The **MMS** option should be used where applicable (e.g. if the port provides local control panel information to MMS) to cause the compiler to output necessary port information to the MMS information file. For specifics on other possible options, refer to the appropriate protocol manuals.
- *port-settings* are records that specify port settings such as baud rate and data format.
- *messages* define the control, indication and special messages for the port.

11.2.2 Port Settings Records

11.2.2.1 OPERATING MODE Records

The OPERATING MODE record is used to determine the electrical mode of serial ports 1 and 2 of the NVSP board. It is an optional record and must be the first non-comment record following the SERIAL PORT record. The 'CONFIGURATION FILE' record should follow this record. This record is only valid on Serial ports 1 and 2 of an NVSP board.

If this record is not entered for port 1 or 2 of an NVSP board, the default electrical mode of the port is RS-232.

The record format is:

OPERATING MODE = *type*

- *type* names the type of electrical connection for Serial ports 1 and 2 of the NVSP board. Valid entries are RS-232, RS-422 or RS-485.

For example:

```
OPERATING MODE = RS-485
```

11.2.2.2 CONFIGURATION FILE Records

This optional record specifies the name of a user-created file containing protocol-specific information to override what is normally supplied by the CAA.

The record format is:

CONFIGURATION FILE = *filepath*

- *filepath* is the path of the user configuration file. If no directory is specified, the file is assumed to be in the same directory as the .CSS file.

For example:

```
CONFIGURATION FILE = D:\CONFIG\K2_USER.LPC
```

11.2.2.3 DEFAULT BAUD RATE Records

This optional record identifies the default baud rate for serial communications on a given port. If the record is not present, baud rate is determined by the CAA or user defined configuration file for the protocol. Default baud rate is used if field wiring is not available to select a baud rate, or if the field wiring indicates the default baud rate is adequate (baud rate control code = 000).

The record format is:

DEFAULT BAUD RATE = *rate*

- *rate* is the decimal number which defines the baud rate. The allowable values for this field are 0, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 and 19200.

For example:

```
DEFAULT BAUD RATE = 1200
```

11.2.2.4 BAUD RATE CONTROL Records

This optional record is available in most protocols to identify a set of input variables whose values determine the baud rate. When present, these values override all other baud rate specifications.

The record format is:

BAUD RATE CONTROL = *name-3, name-2, name-1*

- *name-1, name-2* and *name-3* are the names of the input or logic parameters whose values determine baud rate as described in the application rules for non-vital serial communications.

For example:

```
BAUD RATE CONTROL = NVI-03, NVI-02, NVI-01
```

11.2.2.5 DATA FORMAT Records

The DATA FORMAT data record identifies the serial data format. This is an optional data record used in most protocols. If the record is not present, the default data format identified in the CAA or user defined configuration file is used. This record should only be used if the defaults are not satisfactory.

The record format is:

DATA FORMAT = *databits , stopbits , parity*

- *databits* identifies how many data bits to transmit, 7 or 8. Some message protocols require 8 data bits per byte and this value cannot be overridden.
- *stopbits* is number of stop bits in the transmitted byte, 1 or 2.
- *parity* indicates the parity. This field should be E for even parity, O for odd parity or N for no parity.

For example:

DATA FORMAT = 7,2,E

DATA FORMAT = 8,1,O

DATA FORMAT = 7,1,N

11.2.2.6 ONLINE CONTROL Records

This optional record identifies a variable which, when true, disables message transmission from the serial port.

The record format is:

ONLINE CONTROL = *name*

- *name* is the name of an input or logic variable which, when true, disables transmission of serial messages from the port.

For example:

ONLINE CONTROL = PORT1-XMIT-STOP

11.2.3 Serial Messages

Two methods are provided for defining serial messages and their contents. Either can be used interchangeably.

The first method identifies an incoming message with a CONTROL record, and an outgoing message with an INDICATION record. The compiler processes all such messages encountered.

The second method uses a MESSAGE record to indicate a new message, and SOURCE and DESTINATION records to provide a text name for the message sender and receiver. The compiler processes a message only if the SOURCE and/or DESTINATION name matches the NVSP board ID name provided in the documentation section for the board (.CSI file). This method is most useful in serial communications protocols such as DT8 PEER, where both ends of the serial link may be NVSP boards. A single file can sometimes be provided for all serial links, and data is assigned based on the system currently being compiled.

11.2.3.1 Control (Incoming) Messages

Control messages start with a CONTROL record or a MESSAGE record preceded by a SOURCE and/or a DESTINATION record.

The control record format is:

CONTROL = **ADDRESS***(address)* , **LENGTH***(length)*

1 = control-bit-1

2 = control-bit-2

…

The message record format is:

SOURCE = *source-board-name*

DESTINATION = *dest-board-name*

MESSAGE = **ADDRESS***(address)* , **LENGTH***(length)*

1 = control-bit-1

2 = control-bit-2

…

Messages that start with a CONTROL record are always processed; messages that start with a MESSAGE record are processed only if the board name in the DESTINATION record matches the name in the NVSP ID record for the application that is being compiled.

- *address* is a binary address that identifies the message, 1 to 32 bits. How the message address is used depends on the protocol.
- *length* is the number of bits (variables) in the message.
- *control-bits* are the variables that make up the message. These are received variables which can be used as inputs to non-vital logic, hardware outputs or output bits in other messages. PERMONE or PERMZERO can be used to designate unused message bits. Message bit position numbers must range sequentially from 1 to the defined message length; if all bits are not specified for the full message length, the compiler pads the rest of the message with PERMZERO.

11.2.3.2 Indication (Outgoing) Messages

Indication messages start with an INDICATION record or a MESSAGE record preceded by a SOURCE and/or a DESTINATION record:

The indication record format is:

INDICATION = **ADDRESS***(address)* , **LENGTH***(length)*

1 = indic-bit-1

2 = indic-bit-2

…

The message record format is:

SOURCE = *source-board-name*

DESTINATION = *dest-board-name*

MESSAGE = **ADDRESS***(address)* , **LENGTH***(length)*

1 = indic-bit-1

2 = indic-bit-2

…

Messages that start with an INDICATION record are always processed; messages that start with a MESSAGE record are processed only if the board name in the SOURCE record matches the name in the NVSP ID record for the application that is being compiled.

- *address* is a binary address that identifies the message, 1 to 32 bits. How the message address is used depends on the protocol.
- *length* is the number of bits (variables) in the message.
- *indic-bits* are the variables that make up the message. These are transmitted variables which can be taken from non-vital logic variables (hardware inputs from other messages). PERMONE indicates that a message bit is always True; PERMZERO indicates that a message bit it always False. Array elements can be used as indication message bits.

11.2.3.3 Text Messages

To identify a text message, preface the normal CONTROL, INDICATION or MESSAGE header with the word "TEXT."

The data format is:

TEXT MESSAGE = **ADDRESS**(*address*) , **LENGTH**(*length*) , **NAME**(*name*)

- *address* is a binary address from 1 to 32 bits, as for normal serial messages.
- *length* is the number of characters in the message.
- *name* is an optional message identifying name of up to 16 characters, used when linking text messages.

The compiler processes or ignores the message using the same criteria as for binary messages: it always processes CONTROL and INDICATION messages, and processes MESSAGE messages only if the source or destination board name matches the NVSP name for the application.

An incoming text message can be linked to an outgoing one to provide automatic transfer of text data between ports. Messages must be linked on a one-to-one basis only. Linking is accomplished by using the same message name when specifying text messages on different ports. When this is done, only one of the messages needs a length designation and a separate address designation is required only if the incoming and outgoing message addresses are actually different.

For example:

```
SERIAL PORT 2 = TYPE(DT8)
    TEXT CONTROL = ADDRESS(00001),LENGTH(100),NAME(HHT-OUT)
    TEXT INDICATION = ADDRESS(00010),LENGTH(100),NAME(HHT-IN)
SERIAL PORT 4 = TYPE(DT8)
    TEXT CONTROL = ADDRESS(11110), NAME(HHT-IN)
    TEXT INDICATION = NAME(HHT-OUT)
```

Allows the control message received at port 2 to be transmitted from port 4 with the same address, and the control message received at port 4 to be transmitted from port 2 with a different address.

11.2.3.4 Special Messages

Special messages are defined in a manner similar to ordinary binary serial messages, but using the SPECIAL keyword preceding CONTROL or MESSAGE. Numbered data elements are used to transfer communication event information between the protocol emulation and application sections of the system. Flag parameters are set, either by the application to request that an action be taken by the protocol emulation, or by the protocol emulation to announce the arrival of information. Generally the notified system is responsible for resetting the flag once the requested action has been completed.

The number, size, contents, and usage of special messages are determined by the protocol assigned to the serial port. See the appropriate protocol manual for specifics on using special messages.

The data format is:

SPECIAL CONTROL = **LENGTH** *(length)*

1 = spcl-message-bit-1

2 = spcl-message-bit-2

…

- *length* is the total number of data elements in the special message. This field is required.
- *spcl-message-bits* are the user-assigned names for the data elements. Usage of these elements depends on the protocol assigned to the serial port.

For example:

```
SERIAL COMMUNICATIONS SECTION

SERIAL PORT 1 = TYPE(DT8 SLAVE) , UNLATCHED CONTROLS
DEFAULT BAUD RATE = 9600
ONLINE CONTROL = ON-LINE
DATA FORMAT = 8,1,N
OPERATING MODE = RS-232
CONFIGURATION FILE = Q1_DT8.LPC

CONTROL = ADDRESS(00000100),LENGTH(2)
1 = 112_NZSV-ASI
2 = 112_RZSV-ASI

INDICATION = ADDRESS(00000100),LENGTH(3)
1 = 112ENWK-PTM
2 = 112ERWK-PTM
3 = 112ENJPK-PTM

TEXT INDICATION = ADDRESS(00000100),
LENGTH(1),NAME(MSG0036)

SPECIAL CONTROL = ADDRESS(00000100),LENGTH(24)
1 = PERMZERO
2 = PERMZERO
3 = PERMZERO
4 = PERMZERO
5 = P1_SM5

SERIAL PORT 2 = TYPE(ATCS)
…
```

11.3 NETWORK SERIAL COMMUNICATIONS FILE (.NSS)

This part of the manual covers the CAA inputs required to specify network communications protocols.

Several protocols are available through the CAA. No effort is made in this manual to list all available protocols, or to describe all the rules for their application. This section of the manual lists CAA inputs that are common to most or all protocols. Refer to the appropriate documents for information specific to the actual protocols being used.

The .NSS file consists of a section header followed by one or more network port definitions. A network serial port is defined by its network settings, the protocol it uses, and the related set of serial messages used to communicate with one or more remote network stations.

CAA versions before 610 without subnets/redundancy incorporated network settings such as network devices and remote IP addresses into this file. For CAA versions (iVPICAA 610 and later), the network settings have been moved to separate files. The network serial port data in this file describes the protocol and messages handled by the board's Main Processor; the network settings for the corresponding NVSOE (NonVital Serial over Ethernet) node which is handled by the Comm Processor are described in the NVSOE Links Definition and NVSOE Connection Data files.

The file format is:

NETWORK SERIAL COMMUNICATIONS SECTION

network-port-definition-1

network-port-definition-2

…

The format for user-defined network port definition is:

NETWORK PORT = **TYPE** *(protocol-name)* [**DEVICE***(network-device)*,

port-options]

port-settings

message-1

message-2

…

message-n

- *port* is the network port to receive and transmit messages and is a decimal number from 1 to 10.
- *protocol-name* names the type of code system to be emulated on the port, or protocol for serial communications with another device. See the appropriate protocol manual for the name needed to indicate the desired protocol.
  - *network-device* identifies the network device which will handle the communications, either ENET1 or ENET2. The IP address of this device must be identified in the main compiler file. This field is used only for old CAA versions without subnets/redundancy. CAA versions 610 and later with subnets/redundancy distinguish between the definitions of the network serial ports and their messages which are done in this file and the definition of the NonVital Serial over Ethernet settings that control how the Comm Processor passes message data over the network and are done in other files. For CAA versions 610 and later, the DEVICE field is not used and the NVSOE LINKS DEFINITION and NVSOE CONNECTION DATA sections contain the network settings.
- *port-options* refers to any options specific to a given protocol. Two options are available to all protocols: **UNLATCHED CONTROLS** (application clears all controls for this port once per application cycle) and **LATCHED CONTROLS** (controls are not automatically cleared). The **MMS** option can be used where applicable to make serial port information available to MMS through the MMS information file. For specifics on the use of port options, refer to the appropriate protocol documents.
- *port-settings* are records that specify network port settings.
- *messages* define the control, indication and special messages for the port.

It is also possible to define a network port for communicating local control panel information with MMS:

MAC TCP PANEL PORT = **DEVICE***(network-device)*

control-message

indication-message

- *network-device* is a network device with diagnostics enabled, either ENET1 or ENET2. This field is used only for CAA versions before 610 without subnets/ redundancy. CAA versions 610 and later assign the MAC TCP PANEL PORT to whichever device has been assigned the MAC TCP interface by the MACTCP record in the .CSI file. Also, the DEVICE field is not used.

11.3.1 Network Port Settings Records

11.3.1.1 CONFIGURATION FILE Record

This optional record specifies the name of a user-created file containing protocol-specific information to override what is normally supplied by the CAA.

The record format is:

CONFIGURATION FILE = *filepath*

- *filepath* is the path of the user configuration file. If no directory is specified, the file is assumed to be in the same directory as the .NSS file.

For example:

CONFIGURATION FILE = D:\CONFIG\DT8_USER.LPC

11.3.1.2 ONLINE CONTROL Record

NOTICE

This feature is not currently available for Ethernet Ports. It is disabled.

This optional record identifies a variable which, when true, disables message transmission from the network port. The record format is:

ONLINE CONTROL = *name*

- *name* is the name of an input or logic variable which, when true, disables transmission of serial messages from the network port.

For example:

ONLINE CONTROL = PORT1-XMIT-STOP

11.3.1.3 REMOTE NETWORK CONNECTION Record

This record identifies the network properties of the remote network device. It is available only for CAA versions before 610 without subnets/redundancy, and is required only if this information is required by the protocol. The record format is:

REMOTE NETWORK CONNECTION = **IP***(remote-ip-address)*,
PORT*(remote-port)*

- *remote-ip-address* is the remote IP address
- *remote-port* is the remote Ethernet network port number.

For example:

REMOTE NETWORK CONNECTION = IP(162.21.17.19), PORT(1108)

11.3.2 Messages

Control, Indication, Text, and Special messages are specified as in the Serial Communications (.CSS) file.

For Example

```
NETWORK SERIAL COMMUNICATIONS SECTION

NETWORK PORT 1 = TYPE(DT8 SLAVE) , UNLATCHED CONTROLS
CONFIGURATION FILE = Q1_DT8.LPC

CONTROL = ADDRESS(00000100),LENGTH(2)
1 = 112_NZSV-ASI
2 = 112_RZSV-ASI

INDICATION = ADDRESS(00000100),LENGTH(3)
1 = 112ENWK-PTM
2 = 112ERWK-PTM
3 = 112ENJPK-PTM

SPECIAL CONTROL = ADDRESS(00000100),LENGTH(24)
1 = PERMZERO
2 = PERMZERO
3 = PERMZERO
4 = PERMZERO
5 = P1_SM5
```

11.4 NVSOE CONNECTION DATA FILE (.NNS)

This part of the manual covers the CAA inputs required to describe the network properties of the NonVital Serial over Ethernet (NVSOE) nodes in the network. It is available only for CAA versions with subnets/redundancy.

A NVSOE node corresponds to a network serial port in the .NSS file and is identified by a name constructed by appending the network serial port number to the NVSP board ID and separating the two with a semicolon. For example, if the board name is "MY-NVSP" and the network serial port number is 2, the NVSOE node name will be "MY-NVSP : 2". The maximum size of the NVSOE node name is 40 characters, and if the combined name is too long the board ID portion will be truncated accordingly.

External devices that are not NVSP or equivalent boards are identified by a name without the added port number.

The file consists of a section header followed by one or more NVSOE node definitions.

The file format is:

NVSOE CONNECTION DATA SECTION

node-definition-1

node-definition-2

…

The format for each node definition is:

NVSOE = **NAME** *(node-name)* [, **REDUNDANT**] , **IP***(ip-address)* [, **PORT***(nvsoe-port)* , **MASK***(subnet-mask)*]

- *node-name* is the name of an NVSOE node on a NVSP or equivalent board, entered in "board name : port number" format as described above, or the name of a non-board node. The data for this node will be used in a given application if the node is one of the network serial ports for this board or if it is a remote node linked to one of the network serial ports.
- *REDUNDANT* indicates that communications with this node are path-redundant.
- *ip-address* specifies the IP address(es) of the node. If redundant, two IP addresses separated by a comma are required.
- *nvsoe-port* identifies the Ethernet port used by the node. Available entries are:
  - ENET1-N, ENET2-N, REDUNDANT-N: identifies the network device(s) used by this local or remote node. The CAA automatically assigns port values using the new ports assignment scheme (50000 range). These entries are only available for CAA versions with subnets/redundancy. When available, they are the preferred values unless this node is to communicate with a node in an existing application compiled using a CAA version before 610.
  - ENET1, ENET2: identifies the network device used by this local or remote node. The CAA automatically assigns values using the old ports assignment scheme (1000 range). These entries can be used for CAA versions before 610 with no subnets/redundancy, or if the old port values must be retained for communications with an NVSOE node in an existing application.
  - One or two numeric values depending on whether the node is redundant.
- *subnet mask* designates the mask(s) which, when combined with the IP address data, specifies the subnet(s) on which this node resides. It is available only for CAA versions with subnets/redundancy. If redundant, two masks separated by a comma are required. This field is optional; if it does not exist, default values of 255.255.255.0 will be assigned.

The fields in these records can be placed on multiple lines if each line but the last ends with a comma.

For example:

```
NVSOE CONNECTION DATA SECTION

NVSOE = NAME(MY-NVSP : 1),IP(172.16.21.17),PORT(ENET1-N)

NVSOE = NAME(REMOTE-NVSP : 5),IP(172.16.21.55),PORT(ENET2-N)

NVSOE = NAME(MY-NVSP : 3),REDUNDANT,
    IP(172.16.21.17,172.17.22.17),
    PORT(REDUNDANT-N),
    MASK(255.255.255.0,255.255.255.0)

NVSOE = NAME(REMOTE-DEVICE),REDUNDANT,
    IP(155.20.21.5,156.20.21.5),
    PORT(100,200),
    MASK(255.255.255.0,255.255.255.0)
```

11.5 NVSOE LINKS DEFINITION FILE (.NCW)

This part of the manual covers the CAA inputs required to map the NonVital Serial over Ethernet (NVSOE) links between network serial ports and other devices. It is available only for CAA versions with subnets/redundancy.

An NVSOE node corresponds to a network serial port in the .NSS file and is identified by a name constructed by appending the network serial port number to the NVSP board ID and separating the two with a semicolon. For example, if the board name is "MY-NVSP" and the network serial port number is 2, the NVSOE node name will be "MY-NVSP : 2". The maximum size of the NVSOE node name is 40 characters, and if the combined name is too long the board ID portion will be truncated accordingly.

External devices that are not NVSP or equivalent boards are identified by a name without the added port number.

The file consists of a section header followed by one or more NVSOE link definitions.

The file format is:

NVSOE LINKS DEFINITION SECTION

link-definition-1

link-definition-2

…

The format for each link definition is:

NVSOE LINK = **TYPE** *(protocol-name)* [, **REDUNDANT**]

LOCAL = *local-nvsoe-name* [, **CLIENT** | **SERVER**]

REMOTE = *remote-name* [, **CLIENT** | **SERVER**
GATEWAY*(gateway-name)*, **ID***(remote-id)*]

REMOTE = ...

- *protocol-name* names the type of code system to be emulated or the protocol for serial communications with another device. It must be identical to the protocol for the corresponding network serial port in the NETWORK SERIAL COMMUNICATIONS section. See the appropriate protocol manual for the name needed to indicate the desired protocol.
- **REDUNDANT** indicates that the link is path-redundant.
- *local-nvsoe-name* is the name of the local NVSOE node, entered in "board name : port number" format as described above. The link definition will be used in a given application only if the local name corresponds to one of the network serial ports on the board in question.
- **CLIENT** or **SERVER** specifies whether the local or remote node acts as a client or as a server in the link. If neither is present, the node is neither client nor server. Note that CLIENT and SERVER can describe which node requests a TCP/IP connection as well as whether a node behaves functionally as a client or a server. The rules for using CLIENT and SERVER depend on the protocol.
- *remote-name* is the name of a remote NVSOE node on a NVSP or equivalent board, constructed as described above, or the name of a non-board device.
- *gateway-name* provides the name of the gateway(s) required when the NVSOE link is across subnets. If redundant, two names separated by a comma are required; if only one path of a redundant link crosses subnets, the gateway name for the other path should be NONE. The names in this field must correspond to gateway names in the records of the GATEWAYS SECTION; the IP address of each gateway must be on the same subnet as the corresponding device of the local NVSOE node.
- *remote-id* is an optional numeric identifier that can be given to each remote node in a link. Requirements for its usage depend on the protocol.

The fields in these records can be placed on multiple lines if each line but the last ends with a comma.

For example:

```
NVSOE LINKS DEFINITION SECTION

NVSOE LINK = TYPE(DT8 SLAVE)
LOCAL = MY-NVSP : 1, CLIENT
REMOTE = REMOTE-NVSP : 5, SERVER

NVSOE LINK = TYPE(DT8 SLAVE),REDUNDANT
LOCAL = MY-NVSP : 3, CLIENT
REMOTE = REMOTE-DEVICE, SERVER, GATEWAY(GW-1,GW-2)
```

11.6 DATA LOGGER FILE (.LOG)

This file contains the records required to specify data logging. The DATA LOGGING record must appear in the documentation section of the main input file (.CSI). All other records for data logging are to be placed in the DATA LOGGING SECTION of this file.

The .LOG file consists of a DATA LOGGING SECTION header followed by data logging option records and optional application log messages.

The file format is:

DATA LOGGING SECTION

data-logging-options

application-log-messages

11.6.0.1 DATA PROTECT Records

The DATA PROTECT data record indicates the minimum time period before a record can be erased. This record is required in the Data Logging Section.

The record format is:

DATA PROTECT = *hours* **HOURS**, *minutes* **MINUTES**

- *hours* is a one- to three-digit number, from 0 to 500, specifying the time period in hours.
- *minutes* is a one- or two-digit number, from 0 to 59, specifying the time period in minutes.

To disable the option, use a time period of zero. The default value is 6 hours.

For example:

DATA PROTECT = 24 HOURS

DATA PROTECT = 30 MINUTES

DATA PROTECT = 1 HOUR, 45 MINUTES

DATA PROTECT = 0

11.6.0.2 AUTO DUMP Record

The AUTO DUMP data record specifies if and how data is automatically dumped from the data logger to the MAC port. This record is optional in the Data Logging Section. If the record does not exist, auto dumping is not done.

The record format is:

AUTO DUMP = *dump_mode*

- *dump_mode* is the type of auto dumping required. These are the valid auto dump modes (choose only one):
  - OFF: auto dumping not done.
  - MEMORY END: dump logger contents when data logger memory is full. This is done when any auto dump mode other than "off" is used.
  - NEW DIRECTORY: older directory contents are dumped as soon as a new directory is created.
  - PERIODIC(hrs): dump contents periodically at the time interval "hrs" hours. The maximum time is 255 hours.

For example:

AUTO DUMP = OFF

AUTO DUMP = NEW DIRECTORY

AUTO DUMP = MEMORY END

AUTO DUMP = PERIODIC(8)

11.6.0.3 SYSTEM SNAPSHOT PERIOD Record

This optional record defines the interval between periodic snapshots of the system information, including non-vital inputs and outputs.

The record format is:

SYSTEM SNAPSHOT PERIOD = *hours* **HOURS**, *minutes* **MINUTES**

- *hours* is a one- or two-digit number, from 0 to 24, specifying the time period in hours.
- *minutes* is a one- or two-digit number, from 0 to 59, specifying the time period in minutes.

To disable the option, use a time period of zero, or omit this record. The default value is 0.

For example:

SYSTEM SNAPSHOT PERIOD = 6 HOURS

SYSTEM SNAPSHOT PERIOD = 30 MINUTES

11.6.0.4 DATA LOG Records

These optional records specify the type of data to be logged. Some data types are specific to a particular serial port, while others apply to the entire system.

The record format is:

DATA LOG = **PORT** *port (port-event-list)*

DATA LOG = *(general-event-list)*

- *port* is a serial port number, from 1 to 5.
- *port-event-list* contains one or more event types to be logged for the port. Each event type is separated by a comma, and multiple input lines may be used. These are the valid port-specific event types:
  - CONTROLS: all control messages logged.
  - INDICATIONS: all indication messages logged.
  - BROADCAST: broadcast message from office or master to all field/slave locations.
  - POLL: request for information from another device.
  - CONFIGURATION: request for change in operation settings from terminal/ office.
  - PROTOCOL: message unique to the emulator running.
- *general-event-list* contains one or more event types to be logged. Each event type is separated by a comma, and multiple input lines may be used. These are the valid general event types:
  - INPUTS: all non-vital inputs.
  - OUTPUTS: all non-vital outputs.
  - STATUS: current system status.
  - DIAGNOSTICS: diagnostics message.
  - ERROR: error messages produced by system.
  - SPECIAL: not yet defined.

For example:

DATA LOG = PORT 1 (CONTROLS, POLL, PROTOCOL)

DATA LOG = PORT 2 (CONTROLS, INDICATIONS, BROADCAST)

DATA LOG = (STATUS, DIAGNOSTICS, ERROR)

DATA LOG = (INPUTS, OUTPUTS)

11.6.0.5 INPUT LOG Record

The optional INPUT LOG data record specifies how the non-vital inputs are to be logged, if at all. The non-vital inputs may be logged periodically, filtered, or whenever an input value changes. If the record does not appear, non-vital inputs are not logged.

The record format is:

INPUT LOG = [**PERIOD**(*min* **MINUTES**, *sec* **SECONDS**)] ,
[**SAMPLES**(*samples*)] ,
[**CHANGE DETECT**] ,
[**FLAGGED**]

- *min* is a one- or two-digit number, from 0 to 59, specifying the period in minutes.
- *sec* is a one- or two-digit number, from 0 to 59, specifying the period in seconds.
- *samples* is a one- or two-digit number, specifying the maximum number of changes allowed within the time period PERIOD. The range for this value depends upon the value of PERIOD; the maximum number of samples per second is 3, and the minimum sample rate is 1 sample every 5 minutes. PERIOD data must precede the SAMPLES data.
- CHANGE DETECT indicates the non-vital inputs are to be logged whenever a change in an input's value is detected.

NOTICE

The Change Detect option has been disabled for Input Log.

- FLAGGED is similar to CHANGE DETECT, but the entire non-vital input buffer is stored when an input changes.

For example:

INPUT LOG = PERIOD(1 MINUTE, 30 SECONDS)

INPUT LOG = PERIOD(1 SECOND), SAMPLES(3)

11.6.0.6 OUTPUT LOG Record

The optional OUTPUT LOG data record specifies how the non-vital outputs are to be logged, if at all. The non-vital outputs may be logged periodically, filtered, or whenever an output value changes. If the record does not appear, non-vital outputs are not logged.

The record format is:

OUTPUT LOG = [**PERIOD**(*min* **MINUTES**, *sec* **SECONDS**)] ,
[**SAMPLES**(*samples*)] ,
[**CHANGE DETECT**] ,
[**FLAGGED**]

- *min* is a one- or two-digit number, from 0 to 59, specifying the period in minutes.
- *sec* is a one- or two-digit number, from 0 to 59, specifying the period in seconds.
- *samples* is a one- or two-digit number, specifying the maximum number of changes allowed within the time period PERIOD. The range for this value depends upon the value of PERIOD; the maximum number of samples per second is 3, and the minimum sample rate is 1 sample every 5 minutes. PERIOD data must precede the SAMPLES data.
- CHANGE DETECT indicates the non-vital outputs are to be logged whenever a change in an output's value is detected.
- FLAGGED is similar to CHANGE DETECT, but the entire non-vital output buffer is stored when an output changes.

For example:

```
OUTPUT LOG = PERIOD(10 MINUTES)
OUTPUT LOG = PERIOD(1 SECOND), SAMPLES(2)
OUTPUT LOG = CHANGE DETECT
```

11.6.0.7 PRINT MODE Record

The PRINT MODE record specifies how the events are to be retrieved, either First In First Out (FIFO), or Last In First Out (LIFO).

The record format is:

PRINT MODE = *mode*

- *mode* is either FIFO or LIFO. The default is LIFO.

For example:

```
PRINT MODE = LIFO
PRINT MODE = FIFO
```

11.6.0.8 DATA LOGGING INTERFACE Record

This optional record identifies the source of output Data Logging information.

The record format is:

DATA LOGGING INTERFACE = *type*

- *type* is the interface type: HHT for hand-held terminal, MAC for VT100 or equivalent CRT,

For example:

```
DATA LOGGING INTERFACE = MAC
```

11.6.0.9 DATALOGGER NAMES Record

The optional DATALOGGER NAMES record indicates whether the CAA is to include mnemonic names for user messages in the application PROM code.

The record format is:

DATALOGGER NAMES = *enable*

- *enable* is YES if mnemonic names are to be saved, NO if not.

For example:

```
DATALOGGER NAMES = YES
```

11.6.1 Application Log Messages

Each application log message has this format:

APPLICATION LOG MESSAGE = [**ADDRESS***(addr)*] , **LENGTH***(length)*

1 = log-variable-1

2 = log-variable-2

…

N = log-variable-N

- *addr* is an optional binary address of up to 32 bits identifying this message.
- *length* is the length of the message, the maximum log-variable position from 1 to N.
- *log-variables* are the names of the variables to be logged. PERMZERO and PERMONE are allowed, as are array elements.

For example:

APPLICATION LOG MESSAGE = ADDRESS(1101) , LENGTH(4)

1 = STDBY-IN1

2 = STDBY-STATUS

3 = PERMZERO

4 = BOOLARRAY[4]

11.6.2 Example

LOCATION ID = 1

AUTO DUMP = OFF

PRINT MODE = LIFO

DATA LOGGING INTERFACE = HHT

APPLICATION LOG MESSAGE = ADDRESS(00000100),LENGTH(497)

1 = 112BP-DL

2 = 112EM_DI-DL

3 = 112L-DL

…

11.7 NON-VITAL LOGIC FILES (.PRM/.NV)

These files contain the non-vital logic parameter declarations and the non-vital application logic. A separate .PRM file is needed if array variables are declared and used outside the logic (for example, array elements in output serial messages). The compiler has to know the size of the array before it can evaluate whether an array element has a correct index. Aside from this, the parameter declarations can be moved into the logic file and a separate .PRM file is not needed. The CAAPE always produces a .PRM file from graphics.

The logic file should be the last one processed by the compiler.

Constants and logic parameter declarations must be placed before the application logic.

The file format is:

constant-declarations

Boolean-parameter-declarations

integer-parameter-declarations

timer-parameter-declarations

non-vital-logic-section

11.7.1 Constant Declarations

This section is required if constants are used in the logic. It consists of a CONSTANT DEFINITION SECTION followed by one or more constant definitions.

The record format is:

CONSTANT DEFINITION SECTION

constant-name-1 = value-1

constant-name-2 = value-2

…

- *constant-name* is a unique text name of up to 16 characters that stands for the numeric value.
- *value* is the associated numeric value. Hexadecimal values can be specified by using a “0x” prefix. When the constant name is encountered in the logic, its associated number is substituted. For example, if ARRAY_SIZE = 10, when the array variable declaration:

  ARRAY_VAR[ARRAY_SIZE]
  is encountered the compiler defines ARRAY_VAR to be of size 10.

For example:

CONSTANT DEFINITION SECTION

CONST1 = 123

HEXCONST = 0xFE

11.7.2 Boolean Parameter Declarations

This section is required if there are any internal Boolean parameters. It consists of the section header BOOLEAN PARAMETER SECTION followed by a list of parameter names on one or more lines. Parameter names must be unique, can be up to 16 characters long, and can be separated by spaces or commas. Array declarations are allowed; the array size can be a number or a constant.

For example:

BOOLEAN PARAMETER SECTION

CTC2_PERM0 CTC2_PERM1 CTC2_ON-OFF

PU-DELAY IVPIPWR-UP PWR_UP-OLD

ARRAY_VAR[10]

11.7.3 Integer Parameter Declarations

This section is required if there are any internal Integer parameters. It consists of the section header INTEGER PARAMETER SECTION followed by a list of parameter names on one or more lines. Parameter names must be unique, can be up to 16 characters long, and can be separated by spaces or commas. Array declarations are allowed; the array size can be a number or a constant.

For example:

```
INTEGER PARAMETER SECTION
CYCLE-COUNT, COUNT2, ID_NUMS[50]
```

11.7.4 Timer Parameter Declarations

This section is required if there are any internal Timer parameters. It consists of the section header TIMER PARAMETER SECTION followed by a list of parameter names on one or more lines. Parameter names must be unique, can be up to 16 characters long, and can be separated by spaces or commas. Array declarations are NOT allowed.

For example:

```
TIMER PARAMETER SECTION
CTC2_PUTM BR1-ENTKTM1 BR1-ENTKTM2
```

11.7.5 Non-Vital Logic Section

11.7.5.1 Non-Vital Logic Section / Boolean Equation Section Records

Either record can be placed just after the Boolean, Integer, and Timer Parameter Sections in the non-vital logic file to indicate the start of non-vital logic. The two records are completely interchangeable.

The header record format is:

NON-VITAL LOGIC SECTION
or
BOOLEAN EQUATION SECTION

11.7.5.2 Integer Equation Statements

The statement format is:

result-list = integer-expression

- *result-list* consists of one or more result names separated by commas.
- *integer-expression* consists of integer variables or array elements separated by arithmetic operators.

For example:

```
A,E =VAR1 * (VAR2 + VAR3) +
    VAR4 * (VAR5 + VAR6)
X,Y,Z = 65
M[20] = VALS[X] + 5
```

11.7.5.3 Boolean Equation Statements

The statement format is:

BOOL *result-list = Boolean-expression*

- *result-list* consists of one or more result names separated by commas.
- *Boolean-expression* consists of variables or array elements separated by Boolean operators.

For example:

```
BOOL RES1,RES2 = (A * .N.B +
        C * (D + E))
BOOL PARM1 = FALSE
BOOL X[2] = Y[4] * Z
```

11.7.5.4 TIME DELAY Statements

The statement format is:

TIME DELAY = *delay-spec*

BOOL *timer-result = Boolean-expression*

- *delay-spec* can be a fixed numeric value of HOURS, MINUTES, SECONDS and MSEC, or the name of an integer variable that contains the time delay amount. The delay can occupy more than one line as long as the last symbol on each intermediate line is a comma. Maximum time delay is 6535 seconds in 100-millisecond increments.
- *timer-result* is the name of a timer variable.
- *Boolean-expression* is a standard Boolean expression.

For example:

```
TIME DELAY = 1 HOUR, 10 SECONDS
BOOL TMR = (X * Y)


TIME DELAY = TIMEOUT_VAL
BOOL TMR-2 = (A + B +C)
```

11.7.5.5 APPLICATION Statements

The statement format is:

APPLICATION = section-name

- *section-name* is a descriptive name for the section of logic.

For example:

APPLICATION = ROUTE LOCKING

11.7.5.6 IF..ELSE Statement

The statement format is:

IF(logical-expression)

if-statements-list

ELSE

else-statements-list

- *logical-expression* is a logical expression that evaluates to True or False. The logical expression can occupy more than one line as long as each intermediate line ends in an operator.
- *if-statements-list* is a sequence of one or more statements that are executed if the logical expression evaluates to True. Multiple statements are enclosed in { } brackets.
- *else-statements-list* is a sequence of one or more statements that are executed if the logical expression evaluates to False. Multiple statements are enclosed in { } brackets. The ELSE keyword and the else-statements-list are optional; if they do not exist and the logical expression evaluates to False, the program jumps to the first statement after the if-statements-list.

For example:

```
IF (A==B)
    BOOL C = A*Y+Z

IF (TEST1 && !TEST2)
{
    COUNT = COUNT + 1
    IF (COUNT==10)
            TEST1 = FALSE
}
ELSE
{
    COUNT = 0
}

IF (VAR1)
    BOOL X = X*Y
ELSE IF (VAR2)
    BOOL Y = Y*Z
```

11.7.5.7 WHILE Statements

The statement format is:

WHILE (logical-expression)

statement-list

- *logical-expression* is a logical expression that evaluates to True or False. The logical expression can occupy more than one line as long as each intermediate line ends in an operator.
- *statement-list* is a sequence of one or more statements that are executed as long as the logical expression evaluates to True. Multiple statements are enclosed in { } brackets.

For example:

```
I = 0
WHILE( I<10 )
{
BOOL Y[I] = FALSE
I = I + 1
}
```

11.7.5.8 Statement Labels

The statement format is:

label-name :

- *label-name* is the name of the label.

For example:

```
LABEL:
```

11.7.5.9 GOTO Statements

The statement format is:

GOTO label-name

- *label-name* is the name of the label. When this statement is executed the program finds the label and jump to the statement just after it.

For example:

```
LABELNAME:
N = N+1
…
IF(N<10)
GOTO LABELNAME
```

11.7.5.10 SUBROUTINE Definitions

The data format is:

SUBROUTINE *subroutine-name (argument-definition-list)*

statement-list

[**RETURN**]

END *subroutine-name*

- *subroutine-name* is the name of the subroutine, which is referenced in CALL statements elsewhere in the logic.
- *argument-definition-list* is an optional list of arguments separated by commas. Each argument consists of a type followed by a name. Argument types can be BOOL (passed a Boolean variable address), INT (passed an integer variable address), *BOOL (passed a Boolean value, *INT (passed an integer value). Argument names are used in the statements in the body of the subroutine; their scope is local to the subroutine. The argument list can occupy more than one line as long as each intermediate line ends with a comma.
- *statement-list* is the sequence of statements that are executed when the subroutine is called. Optional RETURN statements can be placed in the statement list to cause an immediate return from the subroutine; if no RETURN statements are encountered, the program returns after the last statement in the list has been executed.

For example:

```
SUBROUTINE X( BOOL A,INT VAL )
BOOL A = TRUE
VAL = VAL = 10
END X
```

11.7.5.11 CALL Statements

The statement format is:

CALL *subroutine-name* [(*argument-list*)]

- *subroutine-name* is the name of the subroutine which has been defined elsewhere in the logic.
- *argument-list* contains one or more Boolean or integer variables or constants and/or logical or integer expressions separated by commas. The number and type of arguments must match those in the definition of the subroutine; if the subroutine has no arguments, the argument list is not needed in the CALL. Since arithmetic operations are allowed in the argument list, logical expressions cannot use the '*' and '+' symbols used in Boolean equations; they use '&&' and '||' instead.

For example:

```
CALL NO-ARGS-SUB
CALL COUNTSUB(TIMED-OUT, COUNT, 5, TRUE)
CALL Y( !FLAG1 && (FLAG2 || FLAG3))
```

11.7.5.12 LIBRARY FILE Records

The record format is:

LIBRARY FILE = *file*

- *file* is the library file name or path.

For example:

```
LIBRARY FILE = D:\LIBFILES\CTA\NORTH.LIB
LIBRARY FILE = LIBFILE
```

11.7.5.13 LIBR Records

The record format is:

LIBR *[filename :] membername [(subst-list)]*

- *filename* is the optional library file name.
- *membername* is the library member name.
- *subst-list* is a list of actual names to be substituted for the generic ones defined for the library member. Multiple input lines are allowed, as long as each preceding line ends with a comma. All generic names in a library member must be given a corresponding actual name; names are assigned in the same order as the member's list of arguments. For example, in LIBR X(A,B,C) A is assigned to the first argument in the argument list, B is assigned to the second and C to the third.

For example:

```
LIBR SWITCH MEMBER 1
LIBR XLIBFILE : MEMBER A
LIBR XLIBFILE : SIGNAL STORAGE( 1WZ, 1XY )
```

SECTION 12 – APPLICATION DATA VERIFIER PROGRAM

12.1 GENERAL

The Application Data Verifier (ADV) Program processes VSP application PROM code data and produces a comprehensive listing so a user can determine whether the PROM code data has been encoded as specified in the compiler input file.

12.2 ADV OVERVIEW

⚠WARNING

ADV INPUT DATA MUST BE VERIFIED SEPARATELY—PRIOR TO ADV PROCESS

Vital system operation requires that the Boolean equations in the Vital application logic must be written correctly, so that by executing the logic, the iVPI system operates safely in accordance with the rules of the transit or railroad authority.

The Application Data Verifier (ADV) output report provides a means to compare and verify equivalence between the input and the output application data.

However, the Application Data Verifier neither determines the safety suitability of the Boolean expression list nor determines the validity of certain encoded iVPI application data. The input data to the ADV process must be verified for safety separately, prior to the ADV process, and the safety and suitability of the input data is the responsibility of the experienced signaling engineer.

The ADV does, however, issue warnings and error messages as a result of non-vital data checking to alert the experienced signaling engineer to possible discrepancies.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

The ADV requires Compiler-generated PROM code data files and a Compiler-generated symbol table file as input. The symbol table file contains the names of parameters assigned by the user in the compiler input file and their corresponding RAM addresses assigned by the Compiler. When the system includes Vital Serial boards there must be a PROM code file for every Vital Serial board in the system.

12.3 CONSOLIDATION REPORTS

The ADV produces a very detailed report, but it is not usually necessary to analyze all of those details. The Consolidation Report files produced by the Compiler and ADV summarize the necessary information for quick checking. See SECTION 12.3 – Consolidation Reports, for more information.

12.4 ADV PHASES

The ADV processes the data in four phases:

1. Input
2. Description Processing
3. Expression Evaluation
4. Checkword Evaluation

An initial graphical logic verification phase occurs in the CAAPE and its results are reported by the ADV.

12.4.1 Input Phase

Input phase involves reading the PROM code and symbol table files into internal memory and prompting the operator for options.

12.4.2 Description Processing Phase

Description processing accounts for the majority of the ADV listing. During this phase, the ADV disassembles the PROM code data and outputs a description of the iVPI hardware configuration and Boolean equation logic. This information must be compared to the compiler input file to determine its validity.

12.4.3 Expression Evaluation Phase

The ADV evaluates all of the terms of the encoded Boolean equations during expression evaluation. The ADV outputs polynomial divider (PD) sums which the user must compare to the MASTER CODELIST and Compiler-generated values to determine their correctness. When Vital Serial boards are in the system the user must compare the generated buffer/memory checkwords, link checkwords, and true/false PD sums with the VITAL SERIAL CODELIST.

12.4.4 Checkword Evaluation Phase

Checkword evaluation phase involves generating main and recheck checkword PD sums which must also be compared against the codelists to determine validity.

12.4.4.1 Graphical Logic Verification

If the logic text files were created from a graphical ladder logic component, the CAAPE compares the equations in the text files against those in the logic component and generates PD sums for both. The results of the comparison are sent to the ADV and reported in the consolidation report file. If the PD sums are identical, this indicates that the logic in the text files is identical to that in the logic component. If they are not, the text files may have been changed or the conversion from graphics to text may have failed.

12.5 ADV PROGRAM

The ADV program is invoked by highlighting an application that contains a valid PROM file (file extension .HEX) or highlighting the PROM file itself and clicking on the ADV toolbar button or selecting the menu option Run | Verify data.

If the system has Vital Serial boards, then the ADV requires one .VBn file for each Vital serial board n in the system. The ADV outputs a listing file with the same name as the input file (file extension .LSV) which consists of 132 characters, variable-length records.

The standard ADV listing (no options) consists of these reports:

SYMBOL TABLE DATA REPORT

DUPLICATE NAMES REPORT

DUPLICATE ADDRESSES REPORT

VITAL INPUT REPORT

VITAL OUTPUT REPORT

* CSEX CONTROLS (CSEX TO VSP) REPORT

BOOLEAN EXPRESSION REPORT

* CSEX SYSTEM INDICATIONS (VSP TO CSEX) REPORT

EXPRESSION EVALUATION REPORT

MEMORY SPECIFICATIONS

TIMER RANGE VALUES

VRD CHECKWORD REPORT

RAM ADDRESS REPORT

DISPLACEMENT/INCREMENT REPORT

SHADOW BANK MEMORY OFFSET DATA

SYSTEM MESSAGE/ERROR REPORT

For non-NVSP systems, the starred reports follow respectively:

CODE SYSTEM CONTROLS/NON-VITAL INPUT REPORT

CODE SYSTEM INDICATIONS/NON-VITAL OUTPUT REPORT

When the system includes one or more Vital Serial boards or defines VSoE communications, five more reports are generated as starred:

SYMBOL TABLE DATA REPORT

DUPLICATE NAMES REPORT

DUPLICATE ADDRESSES REPORT

VITAL INPUT REPORT

VITAL OUTPUT REPORT

CSEX CONTROLS (CSEX TO VPI) REPORT

* VITAL SERIAL INPUT MESSAGE PARAMETERS

* VITAL SERIAL OUTPUT MESSAGE PARAMETERS

BOOLEAN EXPRESSION REPORT

CSEX SYSTEM INDICATIONS (VPI TO CSEX) REPORT

EXPRESSION EVALUATION REPORT

MEMORY SPECIFICATIONS

TIMER RANGE VALUES

* VS MEMORY CONSTRAINTS

VRD CHECKWORD REPORT

RAM ADDRESS REPORT

* VS RAM ADDRESS REPORT

DISPLACEMENT/INCREMENT REPORT

SHADOW BANK MEMORY OFFSET DATA

* VITAL SERIAL TRUE, FALSE, & LINK CODEWORDS

SYSTEM MESSAGE/ERROR REPORT

If the CAA supports DigiSAFE communications and the system includes DigiSAFE messages, two more reports are generated immediately preceding the SYSTEM MESSAGE/ERROR REPORT:

DIGISAFE TRUE, FALSE, & KEY CODEWORDS

DIGISAFE EQUIPMENT ID

These reports displayed are a result of ADV processing options:

ADS PROM CODE DATA REPORT

SYSTEM PROM CODE DATA REPORT

VITAL SERIAL SYSTEM PROM CODE DATA REPORT

VITAL SERIAL DATA STRUCTURES REPORT

In older CAA versions, the following reports may be the result of ADV processing options:

SYMBOL TABLE DATA REPORT

DUPLICATE NAMES REPORT

DUPLICATE ADDRESSES REPORT

RAM ADDRESS REPORT

VS RAM ADDRESS REPORT

- The ADS PROM CODE DATA, SYSTEM PROM CODE DATA, VITAL SERIAL SYSTEM PROM CODE DATA, VITAL SERIAL DATA STRUCTURES, SYMBOL TABLE DATA, DUPLICATE NAMES, and DUPLICATE ADDRESSES REPORTS are (optionally) produced during the input phase.
- The VITAL INPUT, VITAL OUTPUT, CSEX CONTROLS (CSEX TO VPI), VITAL SERIAL INPUT MESSAGE PARAMETERS, VITAL SERIAL OUTPUT MESSAGE PARAMETERS, BOOLEAN EXPRESSION, and CSEX SYSTEM INDICATIONS (VPI TO CSEX) REPORTS are all generated as a result of description processing.
- The EXPRESSION EVALUATION REPORT is produced during expression evaluation processing.
- The MEMORY SPECIFICATIONS, TIMER RANGE VALUES, VS MEMORY CONSTRAINTS, VRD CHECKWORD, RAM ADDRESS, VS RAM ADDRESS, DISPLACEMENT/INCREMENT, SHADOW BANK MEMORY OFFSET DATA, VITAL SERIAL TRUE, FALSE, & LINK CODEWORDS, VITAL SERIAL LINKS AND BLOCKS, DIGISAFE TRUE FALSE & KEY CODEWORDS, and DIGISAFE EQUIPMENT ID REPORTS are generated during the checkword evaluation phase.
- The SYSTEM MESSAGE/ERROR REPORT, produced at the end of ADV processing, contains all messages and/or errors that occurred during processing. In addition, an INDEX to the various reports and their associated page numbers is included on the last page of the ADV listing.

NOTICE

All values given in this manual are for illustrative purposes only and do not necessarily represent valid results of ADV processing.

12.6 INPUT PROCESSING

As the ADV program begins, the user is prompted:
DO YOU WANT TO SPECIFY ANY OPTIONS (Y/N)? ENTER 'X' TO EXIT PROGRAM.

If 'x' is entered, execution terminates. If 'n' is entered, the ADV program continues and produces the reports that comprise the standard ADV listing (described previously). If 'y' is entered by the operator, an options menu is displayed similar to this (actual menu choices depend on iVPI CAA version):

APPLICATION DATA VERIFIER PROGRAM

NOTE: ADV COMPARE DOES NOT REQUIRE ANY OPTION.

1. PRINT ADS PROM CODE FILE DATA.

2. PRINT SYSTEM PROM CODE FILE.

3. PRINT VITAL SERIAL SYSTEM PROM CODE FILES.

4. PRINT VITAL SERIAL DATA STRUCTURES.

5. USE NON-DEFAULT VITAL SERIAL FILE NAMES.

6. ALL OF THE ABOVE

TO TURN AN OPTION ON OR OFF, ENTER NUMBER OF OPTION.

ENTER 'X' TO EXIT PROGRAM.

PRESS <ENTER> TO COMPLETE MENU SELECTION.

Options are toggled on and off until the <Enter> key is entered as the only input. The options selected are indicated by an asterisk. For example, if '3' is entered, an asterisk is displayed next to option 3. Hitting the <Enter> key again selects this option. However, if '3' is entered again, the option is cancelled and the asterisk is removed.

The standard ADV listing (no options) is usually sufficient to verify the PROM code data. The options provide additional information that determines the cause of ADV-generated errors. However, if the ADV displays an option for printing Symbol Table data, this option MUST BE SELECTED in order for the ADV Compare program to be able to process the resulting ADV report file. In CAA versions 25C and later the ADV always produces the Symbol Table data, and no user option is displayed for selecting it.

When no options are selected or after the options are selected ADV lets the user know what type of system it is processing.

When ADV begins main memory processing:

THE SYSTEM IS PROCESSING MAIN MEMORY

For the shadow banks:

THE VSP SYSTEM IS READING SHADOW BANK *n*

- ADV produces a separate message for each shadow bank that it sees.

When ADV begins its processing of main memory for the VSP system:

THE VSP SYSTEM IS READING MAIN MEMORY

The options for selecting non-default Vital Serial file names and for listing VSC PROM code files are currently unused.

12.6.1 ADS PROM Code Data Report

The ADS PROM code data report consists of the PROM code data formatted into columns of 16 words with a 16-bit address field preceding the data. The data format is shown in the example that follows. Both the address and data are displayed in hex. If a PROM code value repeats for one or more lines of output, a single line indicating the address range is produced. This report is useful only if the operator has access to the PROM code data formats.

For example:

ADDRESS ADS PROM CODE FILE DATA

0000 0644 0000 0828 0000 1E8C 0000 1DC6 0000 12E6 0000 2318 0000 137E 0000 1E64 0000

0020 F688 0000 1BE0 0000 07F4 0000 1C2E 0000 1D86 0000 11D2 0000 1EA8 0000 F000 0000

0040 F000 0000 1ED0 0000 0644 0000 0000 0000 1018 0010 1038 0010 1E02 0010 1E22 0010

…

ADDRESS FBE0-FFFF CONTAINS DATA 0000

12.6.2 Symbol Table Data Report

The symbol table data report lists the names, addresses, and buffers of the VSP RAM parameters as they appear in the Compiler-generated symbol table file. Any addresses that are required for expression evaluation that are not assigned names by the user have been assigned names that begin with the '@' character by the Compiler Program.

This is an example of the report format:

BUFFER	ADDRESS	NAME	ADDRESS	NAME
DIN	0180	8T-DI	0184	7RH-DI
	0190	8WP-DI	0194	7RD-DI
	01A0	N8WP-DI	01A4	7LA-DI
	01B0	7RAD-DI	01B4	7LAD-DI
OCK	01C0	@CCK30002-09	01C4	@CCK30002-10
	01D0	@CCK30002-13	01D4	@CCK30002-14
	01E0	7RA-R-CCK	01E4	7RA-R-HCK
	01F0	7LA-R-CCK	01F4	7LA-R-HCK
	0200	7RB-R-CCK	0204	7RB-R-HCK
	0210	7LC-Y-CCK	0214	LC-Y-HCK
	0220	7RA-G-CCK	0224	7RA-G-HCK

Table 12–1 summarizes the buffer names.

Table 12–1. Symbol Table Buffer Names

Buffer Name	Description
DIN	Vital (direct) input parameters
OCK	Output current check (AOCD) parameters, including lamp drive output and output cycle test parameters.
CSC	Code system control parameters and non-vital inputs
VSC	Vital serial controller
DMI	DigiSAFE input parameters (only for CAA supporting DigiSAFE communications)
LA	Self-latched parameters (previous cycle)
LAT	Self-latched parameters (current cycle)
CR	Current result parameters
X	Vital output parameters
TM	Channel 1 Vital timer registers (temporary work area)
LON	LDO 'state' parameters (for protected Vital flashing)
TM2	Channel 2 Vital timers registers

12.6.3 Duplicate Names Report

The duplicate names report provides the names and addresses of all parameters with more than one address. The format for this report is shown in the example. This information is based entirely on the Compiler-generated symbol table file. Only parameters that are self-latched, timer, or LDO 'state' parameters should appear in this report. Timer names always show two duplicates because they appear in the LA, LAT, and TM buffers. ADV uses the address to verify that it is within an allowable buffer type. If ADV sees an address that does not fit within an allowable buffer type, it lets the user know by setting type to *** for that address. The user must verify the name/buffer assignments by matching values against the symbol table.

For example:

DUPLICATE NAME	ADD 1	TYPE	ADD 2	TYPE
7TE	0358	LA	0588	LAT
7RG	0360	LA	0590	LAT
LIGHT1	0368	LA	0598	LAT
LIGHT1	0368	LA	0E3C	TM1
LIGHT1	0368	LA	6555	TM2

12.6.4 Duplicate Addresses Report

The duplicate addresses report provides the addresses and names of all symbol table parameters with the same or overlapping addresses. This information is based entirely on the Compiler-generated symbol table file. No parameters shall have a duplicate address. If any duplicate addresses are found, the ADV aborts processing following the generation of this report. If a duplicate address occurs and this option has not been selected, the ADV prints a message and stops further processing.

For example:

```
ADR 1        NAME 1          ADR 2          NAME 2
NO DUPLICATE ADDRESSES FOUND
```

12.7 DESCRIPTION PROCESSING

During description processing, the ADV extracts data relevant to describing the hardware configuration and Boolean expressions, formats the information, and generates the various reports in the next heading. If the name of a parameter does not have an associated address in the symbol table file, asterisks are produced as the name.

12.7.1 Vital Input Report

The Vital input description report lists the names, addresses, signature header ID, and cycles of forgiveness of each Vital input port (see example). This information is always listed for channel 1 and optionally for channel 2. Inputs are listed in their physical order within each group. If the signature is invalid, a '?' is produced in the signature field.

For example:

CHANNEL 1

SUPERGROUP ADDRESS: 30000 SIGNATURE: A

INPUT GROUP #: 1 ADDRESS: 30006 SIGNATURE: P

(MODULE 2, SLOT 10)

PORT #	NAME	CYCLES
1	SWP	2
2	8T-DI	2
3	8WP-DI	0
4	N8WP-DI	2
5	7RAD-DI	0
6	7RH-DI	2
7	7RD-DI	2
8	7LA-DI	1
9	7LAD-DI	1
10	7LAH-DI	2
11	7LBD-DI	2
12	7LBH-DI	0
13	7LAT-DI	2
14	7LBT-DI	2
15	VRDFRNT-DI	1
16	@DIN30006-16	2

12.7.2 Vital Output Report

The Vital output description report lists the physical port numbers, names, address, data bus assignment (Lo or Hi) and type of each output group: SBO - single break output; DBO - double break output; LDO - lamp drive output; LDO2 - lamp drive output 2; ACO - AC output board. (If the type is invalid, **** are produced as the output type.) If the group contains an output current check parameter, the names of these parameters are also included. For lamp drive outputs hot and cold filament check names are produced, and also flash names, but only if they are encoded in the data structures. In addition, any groups that have been chosen for the output cycle test option have the words **OUTPUT CHECKED** following the board type.

For example:

CHANNEL 1

SUPERGROUP ADDRESS:30000

OUTPUT GP# 1 ADDRESS:30002(HI) TYPE:DBO **OUTPUT CHECKED**

(MODULE 2, SLOT 07)

PT	NAME	CK NAME	FLASH
1	7R-CODE4-DBO	@CCK30002-09	
2	7R-CODE3-DBO	@CCK30002-10	
3	7R-CODE2-DBO	@CCK30002-11	
4	@DBO30002-12	@CCK30002-12	
5	NW-1	@CCK30002-13	
6	NW	@CCK30002-14	
7	RW-1	@CCK30002-15	
8	RW	@CCK30002-16	

OUTPUT GP# 2 ADDRESS:30002(LO) TYPE:LDO
(MODULE 2, SLOT 08)

PT	NAME	HOT CK NAME	COLD CK NAME	FLASH
1	7RA-G-LDO	7RA-G-HCK	7RA-G-CCK	
2	7RA-Y-LDO	7RA-Y-HCK	7RA-Y-CCK	
3	7RD-G-LDO	7RD-G-HCK	7RD-G-CCK	
4	7RD-Y-LDO	7RD-Y-HCK	7RD-Y-CCK	
5	7LA-G-LDO	7LA-G-HCK	7LA-G-CCK	
6	7LA-Y-LDO	7LA-Y-HCK	7LA-Y-CCK	
7	7LB-G-LDO	@HCK30004-15	@CCK30004-15	7G-FL
8	7LB-Y-LDO	7LB-Y-HCK	7LB-Y-CCK	

OUTPUT GROUP# 3 ADDRESS: 3010A(HI) TYPE: ACO
(MODULE 2, SLOT 09)

PT	NAME	CK NAME	FLASH
1	1830-GL-ACO	@CCK3010A-09	
2	1830-GR-ACO	@CCK3010A-10	
3	1834-RD-ACO	@CCK3010A-11	
4	1834-GL-ACO	@CCK3010A-12	
5	1834-GR-ACO	@CCK3010A-13	
6	1836-RD-ACO	@CCK3010A-14	
7	1836-GL-ACO	@CCK3010A-15	
8	1836-GR-ACO	@CCK3010A-16	

OUTPUT GROUP# 4 ADDRESS: 3010C(LO) TYPE: ACO

(MODULE 2, SLOT 10)

PT	NAME	CK NAME	FLASH
1	1840-RD-ACO	@CCK3010C-01	
2	1840-GL-ACO	@CCK3010C-02	840-GL-FL
3	1840-GR-ACO	@CCK3010C-03	1840-GR-FL
4	1842-RD-ACO	@CCK3010C-04	
5	1842-GL-ACO	@CCK3010C-05	
6	1842-GR-ACO	@CCK3010C-06	
7	1844-RD-ACO	@CCK3010C-07	
8	1844-GL-ACO	@CCK3010C-08	1840-GL-FL

12.7.3 NVSP Controls (NVSP to VSP) Report

The Code System Extended Controls report has two messages. The first is eight bits long with only the first bit being currently used. This is the diagnostic message and the bit used is an indication of the NVSP board being active (see example). The second message is the data message.

The sample that follows only shows the first 8 bits of the data message.

For example:

CODE SYSTEM EXTENDED CONTROLS (CSEX TO VSP) DESCRIPTION

MESSAGES FOR CSEX BD 1 (ADDRESS 4000 MODULE 1 SLOT 05) = 2

CSE CONTROL MESSAGE, MSG#: 1CSE CONTROL MESSAGE, MSG#: 2

BIT #	NAME	BIT #	NAME
01	CSEX1_ALIVE	01	4LGZ
02	@CDGO-11-002	02	6LGZ
03	@CDGO-11-003	03	6RGZ
04	@CDGO-11-004	04	3NWS
05	@CDGO-11-005	05	3RWS
06	@CDGO-11-006	06	8DTW
07	@CDGO-11-007	07	1NMS
08	@CDGO-11-008	08	1RWS

12.7.4 Vital Serial Input Message Parameters

This report contains all the message bits and the associated names, including PERMZERO, for every Vital serial input board or VSOE node in the system These messages can vary in length; however, for a multi-drop system it is fixed at 450 bits. Every VS input board or VSOE node has associated with it a fixed diagnostic message of four bits.

Each VSOE input message begins with a CAA-defined parameter related to link data processing. The first bit of the input data message is reserved for message processing and is not available to the application; the user-entered input variables follow. The first bit of the input diagnostic message is the VSOEn-LINKOK application variable.

If the CAA version supports DigiSAFE communications and DigiSAFE messages are configured in the system, this report contains all of the message bits and the associated names, including PERMZERO, for every DigiSAFE input message, by Zone Controller number. The length of each DigiSAFE input message is fixed at 160 bits. No diagnostic messages are associated with DigiSAFE communications.

The sample that follows shows a report of one VS input board, one VSoE message, and an excerpt of a DigiSAFE message.

An example typical short board report for a Vital serial board:

VITAL SERIAL INPUT MESSAGE PARAMETERS BOARD 01

BIT #	NAME	BIT #	NAME
1	4LHR-S	5	8LHR-SI
2	4LDR-SI	6	8LDR-SI
3	6LHR-SI	7	LFR-2WT
4	6LDR-SI	8	GHR-DI

VITAL SERIAL INPUT DIAGNOSTIC PARAMETERS BOARD 01

BIT #	NAME	BIT #	NAME
1	VSC1-ALIVE	3	@VSD0-11-003
2	@VSD0-11-002	4	@VSD0-11-004

VITAL SERIAL INPUT MESSAGE PARAMETERS BOARD 02

BIT #	NAME	BIT #	NAME
1	@VSOE-RLINKOK2	3	REM-DBO-D2
2	REM-DBO-D1		

VITAL SERIAL INPUT DIAGNOSTIC PARAMETERS BOARD 02

BIT #	NAME	BIT #	NAME
1	VSOE-LINKOK		

DIGISAFE INPUT MESSAGE PARAMETERS ZONE CONTROLLER 01

PARAM	NAME	PARAM	NAME
1	REM_DS-IN-6	81	@VSC0-031-081
…			
80	@VSC0-031-080	160	@VSC0-031-160

12.7.5 Vital Serial Output Message Parameters

This report contains all the message bits and the associated names for every Vital serial output board and every VSoE input message in the system, by Vital serial board number.

If the CAA version supports DigiSAFE communications and DigiSAFE messages are configured in the system, this report contains all of the message bits and the associated names, for every DigiSAFE input message, by Zone Controller number. The length of each DigiSAFE input message is fixed at 160 bits.

The sample that follows shows a report of one VS output board and an excerpt of a DigiSAFE message.

For example:

VITAL SERIAL OUTPUT MESSAGE PARAMETERS BOARD 01

BIT #	NAME	BIT #	NAME
1	3WSPR	5	92SWR-DI
2	6RHPR	6	94SWR-DI
3	6RDPR	7	TIMX
4	RPR-2WT	8	RPR-3WT

DIGISAFE OUTPUT MESSAGE PARAMETERS ZONE CONTROLLER 01

PARAM	NAME	PARAM	NAME
1	VRDFRNT2-DI	81	PERMZERO
…			
80	PERMZERO	160	PERMZERO

12.7.6 Boolean Expression Report

The Boolean expression description report consists of all expressions in sum-of-product notation in the order in which they appear in the PROM code file. The expression number (in hex) is a reference to the expressions listed in the Boolean Expression Report, an optional iVPI Compiler output. These hex numbers can be used when comparing expressions. If the PRINT CHANNEL II DESCRIPTION is selected, the channel 2 expressions are alternately listed between the channel 1 expressions.

Normal expressions are decoded and listed using the symbol table names. A '.N.' refers to a parameter in its complement state. Whenever a result name is listed twice for an expression, it must represent a self-latched parameter and '(LAT)' is appended to the name. This also must be verified.

Timer expressions are produced as a three expression set which is preceded by a description of the delay. The encoded timer information for these expressions, although it is used as part of the expression itself, is represented instead as timer ID information. For example, permanently-programmed timers indicate the time delay in minutes and seconds.

Unused output expressions are decoded at the end of this report as 'false' expressions.

This example illustrates the various expression formats:

001C CH1 7LAA = *(normal)*
(7RD * .N.7RAD-DI)

001D CH1 7LA-BTP,7LA-BTP(LAT) = *(self-latched/normal)*
(7LAT-DI * 7LBT-DI)

*** CH1 TIME DELAY = 0 MIN 5 SEC ***
(permanently programmed timer expression set)
001E CH1 @PPVT0011-CR =
(8T-DI)
001F CH1 @PPVT0011-LA,@PPVT0011-LA(LAT) =
(@PPVT0011-CR * @PPVT0011-LA +
@PPVT0011-CR)
0020 CH1 8TP,8TP(LAT) =
(@PPVT0011-CR * @PPVT0011-LA +
@PPVT0011-CR * @PPVT0011-LA * 8TP)

*** CH1 TIMER BOARD 1, TIMER 4 ***
(field-settable timer expression set)
00A4 CH1 @FSTBD1SW4-CR =
(POK-NVI * .N.LIGHT-STK * LIGHT1)
00A5 CH1 @FSTBD1SW4-LA,@FSTBD1SW4-LA(LAT) =
(@FSTBD1SW4-CR * @FSTBD1SW4-LA +
@FSTBD1SW4-CR)
00A6 CH1 LT2,LT2(LAT) =
(@FSTBD1SW4-CR * @FSTBD1SW4-LA +
@FSTBD1SW4-CR * @FSTBD1SW4-LA * LT2)

00A8 CH1 @DBO30002-12 = (FALSE) *(unused output)*

*** CH1 TIME DELAY PROGRAMMABLE = 1 MIN, 45 SEC ***
(field-settable software timer expression set)

```
00B0 CH1 @FSSVT0010-CR =
         (VRDFRNT-DI)
00B1 CH1 @FSSVT0010-LA,@FSSVT0010-LA(LAT) =
         (@FSSVT0010-CR * @ FSSVT0010-LA +
         @ FSSVT0010-CR)
00B2 CH1 TMR1,TMR1(LAT) =
         (@FSSVT0010-CR * @ FSSVT0010-LA +
         @ FSSVT0010-CR * @ FSSVT0010-LA * TMR1)
```

NOTICE

The intermediate timer expression and final timer expression are shown in abbreviated form; however, all address pointers are verified, and thus the display should be sufficient for comparison by the user.

12.7.7 Code System Indications Report

This report contains the CSE board number, indication message number, bit number, and name of the parameter controlling the state of the indication bit for all code system indications, if any. The parameter name can have any of the values listed for the ON parameter in the Non-Vital Output Report.

NOTICE

The NVSP board is often referred to as a Code System Emulator (CSE) board in CAAPE and in reports.

For example:

CODE SYSTEM EMULATOR BOARD #: 2

INDICATION MESSAGE # 1

BIT #	NAME
1	8RWC
2	TRS
3	.N.GK-IND
4	.N.7RAS
5	.N.7LAS
6	.N.7LAH
7	.N.7LA-DI
8	.N.7LAA

12.7.8 NVSP System Indications (VSP to NVSP) Report

This report contains the NVSP board #, the total number of indication messages, and the bit # and name of the parameter controlling the state of the indication bit for all code system indications.

For example:

CSE/CSEX SYSTEM INDICATION DESCRIPTION

CODE SYSTEM EMULATOR BOARD # 1 TOTAL # OF INDICATION MESSAGES: 1

INDICATION MESSAGE # 1

BIT #	NAME	BIT #	NAME	BIT #	NAME
1	1LR	13	3NWCR	25	6RHSR
2	2TPSPR	14	3RWCR	26	0000
3	1ANWCR	15	0000	27	0000
4	1BNWCR	16	0000	28	1ANWP-DI
5	1NWCR	17	3WBR	29	1BNWP-DI
6	1RWCR	18	ET-2TBR	30	1ARWP-DI
7	1ARWCR	19	ET-3TBR	31	1BRWP-DI
8	1BRWCR	20	2WTP-DI	32	3ANWP-DI
9	1WBR	21	ET-2TP	33	3BNWP-DI
10	3LR	22	ET-3TP	34	3ARWP-DI
11	0000	23	4LHSR	35	3BRWP-DI
12	0000	24	6LHSR	36	2E-C5R-DI

12.8 EXPRESSION PROCESSING

During expression evaluation processing, the ADV executes the Boolean equation set as encoded in the PROM code data and generates several PD sums for comparison to the MASTER CODELIST and/or Compiler-generated values by a user.

NOTICE

The ADV has no logic to determine whether these PD sums are correct.

12.8.1 Expression Evaluation Report

The expression evaluation report consists of channel 1 and channel 2 PD sums for expression results, true codewords, false codewords, code system control true codewords, and code system control false codewords. The expression result PD sum is unique to the Boolean equation set and represents the sum of the product terms of all the expressions. This sum is computed by alternating between channel 1 and channel 2 for each product term to guarantee that all product terms are included in the sum. This sum must match the Compiler-generated expression result sum.

True and false codeword PD sums can be compared to the VITAL EXPRESSIONS CODELIST, and are unique for the given number of parameters. The false codeword PD sum includes unused outputs as well as the parameters used in expressions, and so the number of false codewords used is usually a higher number than the number of true codewords used.

The code system control true and false PD sums can be compared to the code system's CODE SYSTEM CONTROLS CODELIST, and are unique for the given number of parameters.

NOTICE

The VITAL EXPRESSIONS CODELIST and CODE SYSTEM CONTROLS CODELIST are included in the ‘ADV Code Lists’ section of the iVPI CAA Reference Online Help.

For example:

	CH1	CH2	#'S
EXPRESSION RESULT PD SUM	FAB7DFC2	8A73610	
TRUE CODEWORD PD SUM	B5EE7AFF	E06AB993	212
FALSE CODEWORD PD SUM	2DED30CA	411B39E4	213
CS CONTROLS TRUE CODEWORD PD SUM	498F166B	4455BFB0	39
CS CONTROLS FALSE CODEWORD PD SUM	0C563079	41927D53	39

12.9 CHECKWORD PROCESSING

During checkword processing, the ADV simulates 'vitally' clearing the iVPI RAM buffers to create main checkwords, and evaluating expressions to generate recheck checkwords. Also, recheck checkwords for lamp drive outputs are sometimes obtained directly from memory. PD sums of each set of checkwords are generated for comparison to the ADV Code Lists by the user. THE ADV HAS NO LOGIC TO DETERMINE WHETHER OR NOT THESE PD SUMS ARE CORRECT.

12.9.1 Memory constraints Report

This report shows that the iVPI system is being instructed to check its full application and system PROM memory space.

Data must always be shown as in this example:

MEMORY CONSTRAINTS

SIZE OF 1 BLOCK 2000 HEX BYTES
FIRST ADDRESS IN ROUTINE 0 0C000 # OF BLOCKS 08
FIRST ADDRESS IN ROUTINE 1 0D000 # OF BLOCKS 08
FIRST ADDRESS IN ROUTINE 2 0E000 # OF BLOCKS 08
FIRST ADDRESS IN ROUTINE 3 0F000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 1 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 2 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 3 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 4 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 5 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 6 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 7 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 8 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 9 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 10 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 11 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 12 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 13 02000 # OF BLOCKS 08
FIRST ADDRESS IN SHADOW 14 02000 # OF BLOCKS 08

FIRST ADDRESS IN SHADOW 15 02000 # OF BLOCKS 08
FIRST ADDRESS IN ADS 02000 # OF BLOCKS 08
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 1
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 2
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 3
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 4
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 5
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 6
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 7
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 8
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 9
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 10
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 11
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 12
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 13
SIZE OF LAST BLOCK IN WORDS 0E00 SHADOW BANK 14
SIZE OF LAST BLOCK IN WORDS 0B80 SHADOW BANK 15
SIZE OF LAST BLOCK IN MAIN ADS 0B80
TOTAL NUMBER OF BLOCKS TO MEMORY CHECK 00A0

12.9.2 Timer Values Report

This report shows that the values in PROM enable the iVPI to correctly verify system timing.

Data must always be shown as in this example:

```
TIMER MINIMUM AND MAXIMUM VALUES


TIMER TMO MIN IS 997504 MAX IS 999551 MICRO SECONDS
TIMER TM2 MIN IS 6656 MAX IS 7167 MICRO SECONDS
TIMER TM3 MIN IS 11776 MAX IS 12287 MICRO SECONDS
TIMER TM4 MIN IS 6656 MAX IS 7167 MICRO SECONDS
```

12.9.3 Vital Serial Memory Constraints

ADV reads each board's data structure to produce a report. This data should be verified by the user to ensure the VS board's PROM entries.

For example:

VITAL SERIAL BOARD 1
SIZE OF 1 BLOCK 0400 HEX BYTES
FIRST ADDRESS IN ROUTINE 00000 # OF BLOCKS 10
FIRST ADDRESS IN ADS 08000 # OF BLOCKS 10

VITAL SERIAL BOARD 2
SIZE OF 1 BLOCK 0400 HEX BYTES
FIRST ADDRESS IN ROUTINE 00000 # OF BLOCKS 10
FIRST ADDRESS IN ADS 08000 # OF BLOCKS 10

VITAL SERIAL BOARD 3
SIZE OF 1 BLOCK 0400 HEX BYTES
FIRST ADDRESS IN ROUTINE 00000 # OF BLOCKS 10
FIRST ADDRESS IN ADS 08000 # OF BLOCKS 10

VITAL SERIAL BOARD 4
SIZE OF 1 BLOCK 0400 HEX BYTES
FIRST ADDRESS IN ROUTINE 00000 # OF BLOCKS 10
FIRST ADDRESS IN ADS 08000 # OF BLOCKS 10

12.9.4 VRD Checkword Report

The VRD Checkword Report consists of main and recheck checkword PD sums and, optionally, the checkwords themselves. Both sums can be compared to the checkword sums in the MAIN CHECKWORD CODELIST and RECHECK CHECKWORD CODELIST to determine validity. The checkwords represent the values used to generate the final sums. These parameters are only meaningful as a diagnostic aid, if the user determines that either the main or recheck sum is invalid.

NOTICE

The MAIN CHECKWORD CODELIST and RECHECK CHECKWORD CODELIST are included in the 'ADV Code Lists' section of the iVPI CAA Reference Online Help.

For example, the main checkword sum and main checkword set:

VRD CHECKWORD REPORT

MAIN CHECKWORD SUM 79F6733B RECHECK SUM 19B7E84B OUTPUTS = 32

Table 12–2. VRD Checkword Report Values

MAIN	SIZE	CHECKWORD	RECHECK CYCLE	CHECKWORD
MCKSUM	19	79F6733B	0	BBE31DC6
CHKIN	16	71F66B3B	1	B2E31546
CHKIB	16	69F6633B	2	A9E30CC6
CHKIA	16	61F65B3B	3	A0E30446
CHKIT	16	59F6533B	4	97E3FBC6
CHKOC	40	51F64B3B	5	8EE3F346
CHKOTC	24	49F6433B	6	85E3EAC6
CHKCS	62	41F63B3B	7	7CE3E246
CHKLA	36	39F6333B	8	73E3D9C6
CHKCR	104	31F62B3B	9	6AE3D146
CHKLAT	36	29F6233B	10	61E3C8C6
CHKX	32	21F61B3B	11	58E3C046

Table 12–2. VRD Checkword Report Values *(Cont.)*

MAIN	SIZE	CHECKWORD	RECHECK CYCLE	CHECKWORD
CHKY	32	19F6133B	12	4FE3B7C6
CHKYN	32	11F60B3B	13	46E3AF46
CHKMEM	0	09F6033B	14	3DE3A6C6
CHKTM0	0	01F6FB3B	15	34E39E46
CKTM12	0	F9F6F33B	16	2BE395C6
CKTM23	0	F1F6EB3B	17	22E38D46
CKTM34	0	E9F6E33B	18	19E384C6
CHKDUM	18	E214E862	19	10E37C46

Buffer size is not meaningful for CHKMEM, CHKTM0, CKTM12, CKTM23 and CKTM34; the report lists buffer sized for these entries as zero (0). Each main checkword is the result of "filling" buffers with "Vital clear" constants, and using the appropriate preconditioning constant to sum the contents. The size of each buffer must be verified by the user as follows to assure that all Vital RAM parameters are "cleared."

See Table 12–3 for buffers and sizes.

Table 12–3. VRD Checkword Report Buffer Names

Buffer	Size
MCKSUM	Main checkword. Size is always 19.
CHKIN	<DIN>, <DIN'> cleared. Size is number of DI ports.
CHKIB	<DINB>, <DINB'> cleared. Size is number of DI ports.
CHKIA	<DINA>, <DINA'> cleared. Size is number of DI ports.
CHKIT	<TEMPI>, <TEMPI'> cleared. Size is number of DI ports.
CHKOC	<OCK>, <OCK'> cleared. Size is total number of output check parameters: 1 per port for SBO/DBO/ACO with output checking, 2 per port for LDO/LDO2 with filament check.
CHKOTC	<CSCTMP> cleared. Size is number of NVSP-to-VSP message parameters, including 8 diagnostics.
CHKCS	<CS>, <CS'> cleared. Size is number of NVSP-to-VSP message parameters, including 8 diagnostics.
CHKLA	<LA>, <LA'> cleared. Size is number of self-latched parameters, including CAA-generated PPVT and FSSVT LA.
CHKCR	<CR>, <CR'> cleared. Size is number of current results, including CAA-generated CR.
CHKLAT	<LAT>, <LAT'> cleared. Same as CHKLA.
CHKX	<X>, <X'> cleared. Size is number of output ports.
CHKY	<Y>, <Y'> cleared. Size is number of output ports.
CHKYN	<YE(n-1)>, <YO(n-1)> cleared. Size is number of output ports.

Those buffer sizes listed as 0 indicate parameters that are ADV constants (they reflect iVPI values which cannot be computed by the ADV). The size of CHKDUM indicates the number of parameters to calculate this checkword and depends on any of the following features being used by the system:

- Flashing
- Serial communications
- DigiSAFE communications, for CAA supporting this option

If none of these features are programmed, the size of CHKDUM is 0 to indicate that a 'dummy' stored checkword is used. A size different from 0 indicates that this checkword is generated as a proof of Vital clearing of the buffers involved in the implementation of the previous features: LONE buffer for flashing, VSIT and VSIA/VSIB buffers for Vital Serial and VSoE, and DMIA buffer for DigiSAFE.

Since CHKDUM can consist of various buffers, text is added after the reporting of the VRD CHECKWORDS to identify the buffers (and buffer lengths) used in calculating CHKDUM.

For example, the ADV might report:

CHKDUM CONSISTS OF	8	LONE ENTRIES
	100	VSITMP ENTRIES
	80	VSIA/VSIB ENTRIES, AND
	333	DMIA ENTRIES

An example of the recheck checkword sum and recheck checkword set is displayed under VRD Checkword Report. The number of recheck checkwords per sum varies based on the number of Vital outputs in the iVPI system. Regardless of quantity, a unique PD checksum is generated for each recheck cycle. These checksums are themselves summed, to generate a single verifiable number which can be compared to the "recheck checksum" number on the MAIN CHECKWORD CODELIST and RECHECK CHECKWORD CODELIST. The number of outputs used to generate these sums is also displayed and must be verified against the total number of outputs in the iVPI system.

12.9.5 RAM Address Report

The RAM address report is a listing of information on Vital clearing of iVPI RAM buffers. The first page of this report lists all uncleared areas in VSP board RAM. This data should be compared against the CPU RAM directory on the next page of the report to verify that the listed areas correspond only to the iVPI buffers not required to be vitally cleared (MISC, OFS1, TM, OFS2, TM', TRE, W-RCHK, W-MAIN, CSITMP, V, STACK).

For example:

RAM ADDRESSES NOT VITALLY CLEARED

1 10000 1027F

2 10AA4 10B0B

3 10D10 10D57

4 1187C 118E3

5 11AE8 11B41

6 12376 12571

7 1258A 1FFFF

The following lists all the buffers in VSP board RAM. The report format is:

RAM BUFFER	START ADDRESS	END ADDRESS
MISC	10000	1027F
DINA	10280	102BF
DINB	102C0	102FF
DIN	10300	1033F
OCK	10340	1049F
CS	104A0	104C3
VSO	11E42	12375
VSIT	104C4	1053F
VSIA	10540	1056F
DMIA1	10570	10AA3
OFS1	10AA4	10B0B
LA	10B0C	10B53
CR	10B54	10B87
LAT	10B88	10BCF
X	10BD0	10D0F
XN	00000	00000
TM	10D10	10D57

YE	10D58	10E97
YEN-1	10E98	10FD7
TEMPI	10FD8	11017
LONE	11018	11037
LON-1	11038	11057
DINA`	11058	11097
DINB`	11098	110D7
DIN`	110D8	11117
OCK`	11118	11277
CS`	11278	1129B
VSO`	11E42	12375
VSIT`	1129C	11317
VSIA`	11318	11347
DMIA2	11348	1187B
OFS2	1187C	118E3
LA`	118E4	1192B
CR`	1192C	1195F
LAT`	11960	119A7
X`	119A8	11AE7
XN`	00000	00000
TM`	11AE8	11B41
YO	11B42	11C81
YON-1	11C82	11DC1
TEMPI`	11DC2	11E01
LONO	11E02	11E21
LONO-1	11E22	11E41
TRE	12376	124B5
W-RCHK	124B6	12521
W-MAIN	12522	12571
CSCTMP	12572	12589
CSITMP	1258A	1258D
V	1258E	12599
STACK	1FC00	1FFFF

As previously stated, recheck checkwords for lamp drive outputs are obtained from memory for recheck cycles 01 and 02. (This prevents the VRD output from dropping when the lamps are changed, due to the method of calculation of cold filament check parameters.) This practice is acceptable for lamp drive outputs, but not acceptable for other output types. Therefore, the names of outputs that stored checkword values are listed by the ADV, and the list must contain only lamp drive output names. This list is not optional and its contents must be verified.

For example:

OUTPUTS WHICH HAVE STORED RECHECK VALUES:

1GE-LDO
1RE-LDO
1YE-LDO
@LDO30104-4
2GE-LDO
2RE-LDO
2YE-LDO
@LDO30104-8

12.9.6 Displacements and Increments Report

This report verifies that data has been generated to ensure that certain iVPI values vary correctly from cycle to cycle.

These values must always be observed:

VPI CPU DISPLACEMENTS AND INCREMENTS

ADS	MD	MD"	MI	MI"	RD	RD"	RI	RI"
XMRADS	A000	0800	0400	00A0	4000	0800	0100	0080
WMADS	A000	0800	0400	00A0				
WRADS					4000	0800	0100	0080
MEMADS		0800						

12.9.7 Vital Serial Displacements and Increments

These are fixed values which the user must verify in the ADV consolidation report (filename.ACR) or the ADV listing (filename.LSV). The SMRADS structure is used for point-to-point VSC but not for multidrop VSC; the MDRADS structure is used for multidrop VSC but not for point-to-point. All values are unused for Code Rate Generator boards and VSoE nodes.

Required values for point-to-point VSC:

ADS	MD"	MI"
VCLADS	0800	00A0
SMRADS	0800	00A0
MDRADS	0000	0000
VSXADS	0800	00A0
MEMADS	0800	00A0

Required values for multidrop VSC:

ADS	MD"	MI"
VCLADS	0800	00A0
SMRADS	0000	0000
MDRADS	0800	00A0
VSXADS	0800	00A0
MEMADS	0800	00A0

Required values for Code Rate Generator and VSoE boards:

ADS	MD"	MI"
VCLADS	0000	0000
SMRADS	0000	0000
MDRADS	0000	0000
VSXADS	0000	0000
MEMADS	0000	0000

This example shows normal point-to-point communications for VSC 1 & 2, multidrop communications for VSC 3 & 4, Code Rate Generator for board 5, and VSoE for board 6:

ADS	BD	MD	MD"	MI	MI"	RD	RD"	RI	RI"
VCLADS	1		0800		00A0				
SMRADS	1		0800		00A0				
MDRADS	1		0000		0000				
VSXADS	1		0800		00A0				
MEMADS	1		0800		00A0				
VCLADS	2		0800		00A0				
SMRADS	2		0800		00A0				
MDRADS	2		0000		0000				
VSXADS	2		0800		00A0				
MEMADS	2		0800		00A0				
VCLADS	3		0800		00A0				
SMRADS	3		0000		0000				
MDRADS	3		0800		00A0				
VSXADS	3		0800		00A0				
MEMADS	3		0800		00A0				
VCLADS	4		0800		00A0				
SMRADS	4		0000		0000				
MDRADS	4		0800		00A0				
VSXADS	4		0800		00A0				
MEMADS	4		0800		00A0				
VCLADS	5		0000		0000				
SMRADS	5		0000		0000				
MDRADS	5		0000		0000				
VSXADS	5		0000		0000				
MEMADS	5		0000		0000				
VCLADS	6		0000		0000				
SMRADS	6		0000		0000				
MDRADS	6		0000		0000				
VSXADS	6		0000		0000				
VSXADS	6		0000		0000				

12.9.8 Vital Serial Offsets and Offset Increments

For point-to-point VSC, these are fixed values which depend on the user block number assignment.

This example shows normal point-to-point communications for VSC 1 & 2, multidrop communications for VSC 3 & 4, Code Rate Generator for board 5, and VSoE for board 6:

ADS	BD	CH	OFFSET	INCREMENT
VSOADS			XCIPT	XOINC
	1	1	FF515FC6	0FAF6039
	1	2	FA7E61BA	F19EDE47
	2	1	FE535FCE	0EAD6031
	2	2	1E62E1C8	0180DE3F
	3	1	FD555FD6	0DAB6029
	3	2	3E66E1D8	2184DE2F
	4	1	FC575FDE	0CA96021
	4	2	4E68E1E0	35865E25
	5	1	FB595FE6	0BA76019
	5	2	5E6AE1E8	45885E1D
	6	1	FA5B5FEE	0AA56011
	6	2	6E6CE1F0	558A5E15
SMRADS			ROFINT	ROFINC
	1	1	FF515FC6	0FAF6039
	1	2	FA7E61BA	F19EDE47
	2	1	FE535FCE	0EAD6031
	2	2	1E62E1C8	0180DE3F
	3	1	00000000	00000000
	3	2	00000000	00000000
	4	1	00000000	00000000
	4	2	00000000	00000000
	5	1	00000000	00000000
	5	2	00000000	00000000
	6	1	00000000	00000000
	6	2	00000000	00000000

ADS	BD	CH	OFFSET	INCREMENT
MDRADS			ROFINT	ROFINC
	1	1	00000000	00000000
	1	2	00000000	00000000
	2	1	00000000	00000000
	2	2	00000000	00000000
	3	1	FD555FD6	0DAB6029
	3	2	3E66E1D8	2184DE2F
	4	1	FC575FDE	0CA96021
	4	2	4E68E1E0	35865E25
	5	1	00000000	00000000
	5	2	00000000	00000000
	6	1	00000000	00000000
	6	2	00000000	00000000
VSXADS			XOFINT	XOFINC
	1	1	FF515FC6	0FAF6039
	1	2	FA7E61BA	F19EDE47
	2	1	FE535FCE	0EAD6031
	2	2	IE62E1C8	0180DE3F
	3	1	FD555FD6	0DAB6029
	3	2	3E66E1D8	2184DE2F
	4	1	FC575FDE	0CA96021
	4	2	4E68E1E0	35865E25
	5	1	00000000	00000000
	5	2	00000000	00000000
	6	1	00000000	00000000
	6	2	00000000	00000000

ADS VSIADS	BD	CH	OFFSET XOFINT	INCREMENT XOFINC
	1	1	FF515FC6	0FAF6039
	1	2	FA7E61BA	F19EDE47
	2	1	FE535FCE	0EAD6031
	2	2	1E62E1C8	0180DE3F
	3	1	FD555FD6	0DAB6029
	3	2	3E66E1D8	2184DE2F
	4	1	FC575FDE	0CA96021
	4	2	4E68E1E0	35865E25
	5	1	FB595FE6	0BA76019
	5	2	5E6AE1E8	45885E1D
	6	1	FA5B5FEE	0AA56011
	6	2	6E6CE1F0	558A5E15

12.9.9 Shadow Bank Memory Offset Data Report

This report verifies that equation data in successive memory banks on the board are offset by a fixed amount to prevent invalid equation processing if a memory bank addressing error were to occur.

Where shadow memory banks are used, the report must show that successive bank offsets increase by four bytes and that the memory from the bank 1 offset up to the offset for each successive bank has been zeroed.

For example:

SHADOW BANK 1 OFFSET ADDRESS = 0644

SHADOW BANK 2 OFFSET ADDRESS = 0648

SHADOW BANK 3 OFFSET ADDRESS = 064C

SHADOW BANK 4 OFFSET ADDRESS = 0000

SHADOW BANK 5 OFFSET ADDRESS = 0000

SHADOW BANK 6 OFFSET ADDRESS = 0000

SHADOW BANK 7 OFFSET ADDRESS = 0000

SHADOW BANKS 4 TO 7 DO NOT CONTAIN EXPRESSION RESULTS

12.9.10 Vital Serial Report Summary

The Vital serial report summary consists of up to five parts: PD sum values, VSC buffer and memory checkwords, link checkwords, drop address values, and link and block numbers assignments.

12.9.10.1 PD Sum Values

The first part of this report lists the true and false PD sum values as calculated from different Vital Serial Data Structures (SMRADS, MDRADS, VSOADS and VSIADS) for both channels for every Vital Serial board and VSoE communications defined in the system. VSIADS and VSOAD are the CPU input and output data structure.

The example below shows the false PD checkword sums for a system including a CRG board and defining two VSoE nodes. Sums related to the SMRADS and MDRADS structures have been removed from the report. All those sums were zeros since the structures are not used by the example.

For example:

VITAL SERIAL REPORT SUMMARY

TRUE-PDSUMS

<VSOADS>	CPU OUTPUT MESSAGES	BD	CH	CNT	PD-SUM
		1	1	80	95A7E327
		1	2	80	8CF92FA2
	BERT	2	1	5	28AB5B27
	BERT	2	2	5	ADABB694
	ERNIE	2	1	5	49B72FAE
	ERNIE	2	2	5	E8A609A9
	BERT	3	1	3	4DCD75DB
	BERT	3	2	3	BD41A35B
	ERNIE	3	1	3	0CC505C7
	ERNIE	3	2	3	269F6B59

<VSIADS>	CPU INPUT MESSAGES	BD	CH	CNT	PD-SUM
		1	1	0	00000000
		1	2	0	00000000
	BERT	2	1	9	C92626EF
	BERT	2	2	9	C40BA720
	ERNIE	2	1	9	842EBC3D
	ERNIE	2	2	9	4086291F
	BERT	3	1	3	0931B1A9
	BERT	3	2	3	F70BAEA4
	ERNIE	3	1	3	4839C1B5
	ERNIE	3	2	3	6CD566A6

FALSE-PDSUMS

<VSOADS>	CPU OUTPUT MESSAGES	BD	CH	CNT	PD-SUM
		1	1	80	E367FDEF
		1	2	80	663BDF5D
	BERT	2	1	5	6E1981A9
	BERT	2	2	5	81831149
	ERNIE	2	1	5	0F05F520
	ERNIE	2	2	5	C48EAE74
	BERT	3	1	3	F3A5D781
	BERT	3	2	3	39B40E6A
	ERNIE	3	1	3	B2ADA79D
	ERNIE	3	2	3	A26AC668

<VSIADS>	CPU INPUT MESSAGES	BD	CH	CNT	PD-SUM
		1	1	8	61AEA9DD
		1	2	8	8C16E957
		2	1	9	28C7017C
		2	2	9	141CBC90
		3	1	3	B75913F3
		3	2	3	73FE0395

These sums are validated by comparison with the CAA report (LVC) file, in its Vital Serial Summary Sums section.

12.9.10.2 VSC Buffer and Memory Checkwords

The next part of this report lists the VSC buffer and memory checkwords. The first entry, CHKVSO, proves that the Vital serial output (VSO) buffer is cleared. In CAA versions 31746-011 and later, CHKVSI is incorporated into another value and is not reported here. CHKMEMX is the transmit PROM memory check and CHKMEMR is the receive PROM memory check. The next four checkwords, CHKPRX, CHKSOX, CHKSIR, and CHKSRV, are VSC buffer clearing PD-sums. These four must have the same value for every VSC.

These checkwords are validated by comparison with the CAA report (LVC) file, in its Vital Serial Summary Sums section.

The example below shows the report for a system including a CRG board and defining two VSoE nodes.

For example:

```
VITAL SERIAL BUFFER/MEMORY CHECKWORDS
CHKVSO          D1F6CB3B
CHKMEMX         C9F6C33B
CHKMEMR         C1F6BB3B

CHKPRX      BD      CHECKWORD
            1       00000000

CHKSOX      BD      CHECKWORD
            1       00000000

CHKSIR      BD      CHECKWORD
            1       00000000

CHKSRV      BD      CHECKWORD
            1       00000000
```

12.9.10.3 Link Checkwords

The next part of this report shows the link checkwords. The first four link words prove the CPU transmit link (SCXKEY), the VSC receive link (LNKRKEY), the VSC transmit link (LNKXKEY), and the CPU receive link (DCRKEY). These link checkwords are found in the VITAL SERIAL LINK CODELIST. To verify these checkwords the .CW file must be used to obtain the proper LINK numbers for each Vital Serial board or VSoE message. The links are different for transmit and receive operations.

NOTICE

The VITAL SERIAL LINK CODELIST is included in the 'ADV Code Lists' section of the iVPI CAA Reference Online Help.

When VSoE messages are defined, this section of the report also includes the RLinkOK and TLinkOK values for each channel. These values are not codewords and can be verified by comparison with the CAA report (.LVC) file in its section Vital Serial Summary Sums.

For example:

LINK CHECKWORDS	BRD	CH1	CH2
SCXKEY	1	01F3E04B	D19ADE57
LNKXKEY	1	00000000	00000000
LNKRKEY	1	00000000	00000000
DCRKEY	1	00000000	00000000
XMTCHK	1	00000000	00000000
SCXKEY	2	3881E183	843080C3
LNKXKEY	2	00000000	00000000
LNKRKEY	2	00000000	00000000
DCRKEY	2	C97DDE74	9BD2BF34
XMTCHK	2	00000000	00000000
SCXKEY	3	26BDE173	642C80B3
LNKXKEY	3	00000000	00000000
LNKRKEY	3	00000000	00000000
DCRKEY	3	D741DE84	7BCEBF44
XMTCHK	3	00000000	00000000

	BOARD #	LINK #	CH1	CH2
RCV RLINKOK (BERT)	2	1	5D658041	D7458041
RCV RLINKOK (ERNIE)	2	1	5D250043	D3450043
RCV RLINKOK (BERT)	3	3	4D4580C1	D565C0C1
RCV RLINKOK (ERNIE)	3	3	4D0500C3	D16540C3

	BOARD #	LINK #	CH1	CH2
XMIT TLINKOK (RCV 1,BERT)	2	2	BAAA8081	AA8AC081
XMIT RLINKOK (RCV 1,BERT	2	2	45558081	5575C081
XMIT TLINKOK (RCV 1,ERNIE)	2	2	BAEA0083	AE8A4083
XMIT RLINKOK (RCV 1,ERNIE	2	2	45150083	51754083
XMIT TLINKOK (RCV 1,BERT)	3	4	8ACA8101	ACEA0101
XMIT RLINKOK (RCV 1,BERT)	3	4	75358101	53150101
XMIT TLINKOK (RCV 1,ERNIE)	3	4	8A8A0103	A8EA8103
XMIT RLINKOK (RCV 1,ERNIE)	3	4	75750103	57158103

12.9.10.4 Drop Address Values

All values in this report are blank, because it summarizes the multidrop Vital serial data, which is a function not used by the iVPI.

12.9.10.5 Vital Serial Link and Block Assignments Report

For CAA versions included in CAAPE 007 and later, the ADV decodes the application data structures to report the vital serial link and block numbers that were entered by the user and processed by the compiler to assign link keys and message parameter codewords. The values in this report must be compared to the value originally entered in the compiler input files to verify that they are the same; the link and block assignments for this system and for any other systems that communicate with this one via vital serial or VSOE messages must be analyzed to verify that the link and block values were selected in accordance with the rules described in the Vital Serial Links and Blocks application rules section.

A sample report follows; the transmit blocks section of the report has been truncated due to lack of space. Block numbers are reported in the block.sub-block format for comparison to user input, and also as an absolute range of sub-blocks for ease in checking for overlaps. The sub-block ranges show the contiguous codeword assignments in 10-parameter groups, where 001 is the minimum value. For example, in the sample below the receive sub-block range of 098-100 indicates that sub-blocks 098, 099 and 100 were assigned to the receive message. Three sub-blocks were required to provide for a message size of 21 with a sub-block size of 10 parameters.

For example:

VITAL SERIAL LINK AND BLOCK ASSIGNMENTS

BD	CH	TYPE	R LINK	R SIZE	R BLOCK,RANGE	X LINK	X SIZE	X BL..
1	1	VSC	5	21	05.00 098-100	6	15	06.00..
1	2		5	21	05.00 098-100	6	15	06.00..
2	1	MVSC	7	450	10.05 161-205	8	450	12.10..
2	2		7	450	10.05 161-205	8	450	12.10..
3	1	VSOE	1	2	01.02 022-022	2	3	02.00..
3	2		1	2	01.02 022-022	2	3	02.00..

12.9.11 DigiSAFE Report Summary

This report only exists for CAA versions supporting DigiSAFE communications if DigiSAFE messages are defined in the system.

This report lists the true and false PD sum values as calculated from the DMIADS and DMOADS structures for input and output messages, for both channels, and for every Zone Controller (remote DigiSAFE node). The number of vital parameters in each DigiSAFE message is fixed at 160.

The values summed in the PD sum are not codewords but the DigiSAFE transitional values (the values used to convert between the Nisal True and False used by iVPI and their contribution to the message checksum). The only exception is the False PD sum for input messages, where the Nisal False value of each parameter is also included.

This report also lists the SCXKEY and DCRKEY that are general purpose codewords used by iVPI to encode and decode DigiSAFE messages.

The PD sum values and the SCX and DCR keys are only reported for information and cannot be validated. They are however constituent of the DigiSAFE PD sum computed and reported by the CAA and the ADV in their respective consolidation reports.

For example:

DIGISAFE REPORT SUMMARY

TRUE-PDSUMS

<DMIADS> DGSAFE IN MESSAGES	ZC	CH	CNT	PD-SUM
	1	1	160	F9080A85
	1	2	160	231257E8
	2	1	160	B8F3CBAC
	2	2	160	F20E5AC6

<DMOADS> DGSAFE OUT MESSAGES	ZC	CH	CNT	PD-SUM
	1	1	160	2C599630
	1	2	160	E0C27566
	2	1	160	D5B96E66
	2	2	160	35640D71

FALSE-PDSUMS

<DMIADS> DGSAFE IN MESSAGES	ZC	CH	CNT	PD-SUM
	1	1	320	F5C8F7FC
	1	2	320	1C25DB5B
	2	1	320	B7BF2898
	2	2	320	B7356610

<DMOADS> DGSAFE OUT MESSAGES	ZC	CH	CNT	PD-SUM
	1	1	160	82F323BA
	1	2	160	2C60B23E
	2	1	160	82F323BA
	2	2	160	2C60B23E

DIGISAFE CHECKWORDS	ZC	CH1	CH2
SCXKEY	1	125CF0F5	22E40B87
DCRKEY	1	E1A4CF12	DF3A7480
SCXKEY	2	1058F0E5	C0D84B77
DCRKEY	2	E7A8CF22	FF3E7490

12.9.12 DigiSAFE Equipment ID Assignments

This report only exists for CAA versions supporting DigiSAFE communications if DigiSAFE messages are defined in the system.

For CAA versions supporting DigiSAFE communications, the ADV decodes the DigiSAFE application data structures (DMIADS and DMOADS) to report the local (iVPI) and remote (Zone Controllers) Equipment ID numbers that were input by the user to the compiler. The values in this report must be compared to the value originally entered in the compiler input files to verify that they are the same. The Equipment ID assignments for this system and for any Zone Controllers that communicate with this one via DigiSAFE messages should be analyzed to verify that the Equipment ID values are unique to each equipment. These ID assignments should also be analyzed to verify they are unique to all other equipment on the network; this requires a comparison of all Equipment ID numbers listed in the ADV report of all iVPI applications.

For example:

DIGISAFE EQUIPMENT ID ASSIGNMENTS

ZC	TYPE	SOURCE ID	DESTINATION ID
1	IN	1234	11
1	OUT	11	1234
2	IN	3456	11
2	OUT	11	3456

12.10 ADV SYSTEM MESSAGES / ERRORS

As ADV is running it attempts to let the user know what it is processing, general information messages about the system, and of course error messages.

For example, some of the processing messages are:

DESCRIPTIVE REPORT GENERATION BEGINS

EXPRESSION EVALUATION REPORT GENERATION BEGINS

VRD CHECKWORD REPORT GENERATION BEGINS

etc...

For example, some system messages are:

THE SYSTEM HAS A VSP BOARD

THE SYSTEM HAS ONE BANK OF SHADOW MEMORY

SOFTWARE REVISION SIGNATURE xx SITE ID zzzz

PROTECTIVE FLASHING IS IN EFFECT

etc....

12.10.1 Abort and Error Messages

ADV has two types of error messages, one is labeled as an ERROR and ADV continues processing, the other is labeled as an ABORT and ADV stops further processing. Both of these messages have the same format.

The message format is:

Routine Type Message

- where *Routine* tells the user what module found the error, *Type* tells the user how serious the error is (WARNING, ERROR or ABORT), and *Message* is the message ADV is relating to the user.

For example:

Routine	Type	Message
GETCMD	WARNING	SYMBOL TABLE NOT FOUND - ADV GENERATES SYMBOLS[1]
EXPRNM	WARNING	EXPR# z RSLT n & n HAVE SAME NAME nm
GETADS	ABORT	INVALID PROM CODE FILE FORMAT. RECORDS READ n
GETADS	ABORT	INVALID CHECKSUM IN RECORD
EXNORM	ERROR	EXPRESSION RESULT ADDRESS z NOT IN <RAM> BFR
PMOCK	ERROR	ADDRESS NOT CONTIGUOUS:OCK CH n ADDRESS z

1. The ADV does not fully support the feature to generate symbols without a Compiler-generated symbol table and may produce errors during ADV run time.

SECTION 13 – ADV CONSOLIDATION REPORTS

ADV Consolidation report files are generated by the iVPI CAA Vital Compiler ('name'.VCR) and by the ADV ('name'.ACR). The ADV Consolidation reports are broken into 22 sections. The checks performed in each section to verify the correctness of the compiled data structures are described in this section.

NOTICE

The 22nd section [Section 22: DIGISAFE REPORT SUMMARY] is only created by CAA versions supporting DigiSAFE communications.

An ADV Consolidation Check List is provided in Appendix A to record all necessary process steps required to ensure Vital application data structures are correct and validate information contained in the iVPI application before beginning revenue service.

The general theory of operation behind the consolidation reports is that the Vital Compiler and ADV programs both diversely and independently create summaries of their Vital application data: the Compiler lists what was intended to be stored, and the ADV lists what was actually stored in the EPROM files. These summaries are produced using NISAL techniques so that the multitude of application details (Vital board types, addresses and contents; logic equation structures and variable names; etc.) can be reduced to a small set of numeric summary values. The user compares the summary values between the .VCR and .ACR files; if the values are the same, the stored data in the EPROM files matches what was intended by the Compiler. Using NISAL techniques to generate the values ensures that there is a vanishingly small probability that correct matching summary values could be generated using incorrect data.

In the cases where application data cannot be reduced to simple numeric summary values, the data items are listed directly for verification by the user.

⚠WARNING

FSSVT MODIFICATIONS MUST BE VERIFIED

All FSSVT modifications are safety-critical and must be verified, using the AlsDload program or the Application Data Verifier program within CAAPE, to determine whether the iVPI application PROM code data has been encoded as specified by the AlsDload FSSVT compiler.

Refer to Alstom Publication P2521A iVPI Product Overview Manual sections:

Application Verification:

The basis of the application of iVPI is to use a tool to configure the system hardware and software, as well as create the signaling logic for the vital application. The independent Application Data Verifier Tool, as well as associated procedures, must be run and performed prior to any iVPI application program be tested in field commissioning tests.

Proof of Logic (Primordial Logic Review):

The application of iVPI depends on experienced signaling engineers defining configurations and logic to be implemented for the interlocking application. While iVPI guarantees that logic and outputs, etc. are managed vitally, there is no intrinsic check on the correctness or completeness of the signaling logic as it is intended to meet the requirements of the transit or railroad application. It is a primary safety requirement that the logic produced for iVPI execution be independently verified as correct and complete through a "circuit check" type process. The check process must be performed by engineers knowledgeable in the requirements of the signaling rules that govern transit/railroad operation and independent from the engineering staff that produced the logic.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

PROGRAMMING VSP BOARD OVERWRITES FSSVT SETTINGS

Programming an application into a VSP board erases and overwrites the previous application including all FSSVT settings. Any previous field updates to FSSVT settings will be overwritten and the FSSVT settings will be configured per the programmed application.

Failure to monitor and oversee these FSSVT values are as desired can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

FSSVT MODIFICATIONS MUST BE FIELD TESTED

All changes made to the FSSVT must be field tested to validate the intended timer values of any modified timers are observed to be correct in actual operation prior to the return of revenue service.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

FSSVT PASSWORDS MUST BE PROTECTED

FSSVT passwords shall be provided only to responsible personnel that have been properly trained in the FSSVT modification, verification, and validation process.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

FSSVT SIGNATURE VALUES MUST BE VERIFIED

Verify through Vital signatures that FSSVT values that were not intentionally changed have retained their original signature values.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

INTENDED SAFE FUNCTIONALITY OF THE IVPI SYSTEM MUST BE VERIFIED

The safety of the application logic as written is the responsibility of an experienced signaling engineer—CAAPE does not make any determination regarding the inherent safety of the logic equations that were entered.

Verifying the accuracy with which CAAPE converted the experienced signaling engineer's application data into PROM data structures is aided by CAAPE, but the signaling engineer must make a final determination using information supplied by CAAPE. CAAPE’s compilers are not themselves Vital programs. An

additional independent process is needed to verify that the compile was done correctly. This process is required for all Vital applications.

An experienced signal engineer must verify the safety of the iVPI data and its application. It is the signaling engineer's responsibility to verify the correctness of the iVPI input data in that it accurately represents the intended safe functionality of the iVPI system. Furthermore, "verify the correctness" means that the signaling engineer (1) is required to compare the input and output data files to verify the CAA has operated correctly and (2) must test the iVPI application in its intended environment before it can be placed in revenue service.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

IVPI APPLICATION MUST BE VALIDATION TESTED

Prior to revenue service, validation testing must confirm all iVPI application logic is correct and consistent with application requirements.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

IVPI APPLICATION MUST BE FIELD TESTED

Field testing of a iVPI application is required before placing the location into revenue service. The customer’s testing plan and safety plan define the testing requirements for the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

ADV INPUT DATA MUST BE VERIFIED SEPARATELY—PRIOR TO ADV PROCESS

Vital system operation requires that the Boolean equations in the Vital application logic must be written correctly, so that by executing the logic, the iVPI system operates safely in accordance with the rules of the transit or railroad authority.

The Application Data Verifier (ADV) output report provides a means to compare and verify equivalence between the input and the output application data.

However, the Application Data Verifier neither determines the safety suitability of the Boolean expression list nor determines the validity of certain encoded iVPI

application data. The input data to the ADV process must be verified for safety separately, prior to the ADV process, and the safety and suitability of the input data is the responsibility of the experienced signaling engineer.

The ADV does, however, issue warnings and error messages as a result of non-vital data checking to alert the experienced signaling engineer to possible discrepancies.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

VERIFIER MUST BE DIFFERENT THAN DESIGNER

The experienced signaling engineer responsible for verification (the Checker or Verifier) using the ADV checklist and creating the report shall be independent from the signaling engineer responsible for designing (the Designer) the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

THE CONSOLIDATION REPORTS DO NOT LIST POSSIBLE ERROR MESSAGES THAT MIGHT BE REPORTED BY THE ADV. THE ADV REPORT (.LSV) FILE SHOULD ALWAYS BE EXAMINED FOR ERROR MESSAGES BEFORE THE CONSOLIDATION REPORTS ARE USED.

Section 1: SYMBOL TABLE REPORT

No Checks Required.

Section 2: DUPLICATE NAMES REPORT

Timer and self-latched parameters are alphabetically sorted and a PD-SUM is calculated.

Duplicate Name PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 3: DUPLICATE ADDRESS REPORT

There must be no duplicate addresses reported in the ADV report ('name'.ACR).

Section 4: VITAL INPUT REPORT

For each Vital Input board, a PD-SUM is calculated based on supergroup address, supergroup signature, board address, board signature, module, slot, port address, and cycles of forgiveness. In addition, each supergroup signature and board signature is reported for manual verification of uniqueness.

The Vital Input PD-SUMs and the supergroup and board signatures reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical. All supergroup / board signature combinations must be unique.

Section 5: VITAL OUTPUT REPORT

For each Vital Output board, a PD-SUM is calculated based on supergroup address, board address, module, slot, port address, check name address, hot check address, cold check address, and flash name address.

The Vital Output PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

If the CAA report (‘name’.VCR) contains a Vital Output Boards table, verify each SIG PROM value is unique in the table.

If the CAA Report does not contain this table, refer to the .LVC file, Board Report Sections and verify the SIGNATURE PROM for each board is unique.

Section 6: VITAL SERIAL COMMUNICATION CONFIGURATION

No consolidation report is generated for this section.

Section 7: VITAL SERIAL REPORT SUMMARY

For each Vital Serial channel, a PD-SUM is calculated based on the vsoads, vsiads, smrads, mdrads, chkvso, chkmemx, chkmemr, chkprx, chksox, chksir, chksrv, scxkey, lnkxkey, lnkrkey, dcrkey, xmtchk and drop address codewords.

The ADV also examines the data structures to determine which Vital Serial or Code Rate Generator system software files have been selected and reports the corresponding system software part numbers for all boards.

The Vital Serial Summary PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

The reported software numbers should be the same as the software numbers originally entered in the CRG n SOFTWARE records in the VPC file and the VSOE n ID records in the VNT file. Where consolidated lists of Vital Serial Controllers are shown, the order is CRG then VSOE.

Where applicable, the link and block values reported in VITAL SERIAL LINK AND BLOCK ASSIGNMENTS must be the same as those that were originally entered as compiler inputs. Link and block assignments for all interconnected systems must be made in accordance with the rules described in the Vital Serial Links and Blocks application rules section.

Section 8: VITAL SERIAL INPUT MESSAGE PARAMETERS

Vital Serial Input parameters are sorted and a PD-SUM is calculated.

Vital Serial Input PD-SUM reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 9: VITAL SERIAL OUTPUT MESSAGE PARAMETERS

Vital Serial Output parameters are sorted and a PD-SUM is calculated.

The Vital Serial Output PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 10: BOOLEAN EXPRESSION REPORT

The ADV Consolidation routine takes the complex Boolean equations from the Vital logic source file and produces sum-of-products equations. A PD-SUM is calculated based on the sorted parameters of each expanded equation. A PD-SUM is also calculated based on the sorted parameters from the ADV expressions.

This section also includes verification of the complex Boolean equations against the CAAPE ladder logic component that was used to create them. If a ladder logic component was used to create the logic equations (graphical Make Files operation), the CAAPE converts the logic equations back to ladder logic format and compares them against the original component. PD-SUMs are calculated for both. The results of the comparison and the PD-SUMs are passed to the ADV to be reported here.

Both PD-SUMs must be identical.

If the logic equations were created from a graphical ladder logic component, the PD-SUMs reported in this section must be identical. If the PD-SUMs are not identical, this may indicate that either the ladder logic component or the text file has been edited after the Make Files operation, or that an error occurred in the graphics conversion process. Graphical dependency information in the .CFG file output by the compiler should indicate if the text file was edited after graphics conversion.

If the logic equations were entered directly as text, this section reports that there were no graphical dependencies to check.

Section 11: EXPRESSION EVALUATION REPORT

A PD-SUM is calculated based on expression result, channels 1 and 2 codewords and channels 1 and 2 code system control codewords.

Expression Evaluation PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 12: VRD CHECKWORD REPORT

A PD-SUM is calculated based on main and recheck values.

VRD Checkword PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 13: CHECKWORD SIZE

The size of each main checkword is reported in both reports.

Checkword sizes reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 14: EXPLANATION OF CHECKWORD SIZE

No consolidation report is generated for this section.

Section 15: DISPLACEMENT AND INCREMENTS REPORTS

VPI CPU/PD Section:
The values in Table 13–1 must be reported for VSP if Vital Serial is used:

Table 13–1. VPI CPU/PD Section Values

ADS	MD	MD"	MI	MI"	RD	RD"	RI	RI"
XMRADS	A000	0800	0400	00A0	4000	0800	0100	0080
WMADS	A000	0800	0400	00A0				
WRADS					4000	0800	0100	0080
MEMADS		0800						

Vital Serial Section:
The values in Table 13–2 must be reported for point-to-point VSC. The values in Table 13–3 must be reported for multidrop VSC. The values in Table 13–4 must be reported for Code Rate Generator boards.

Table 13–2. VSC Section Values for Point-To-Point VSC

ADS	MD"	MI"
VCLADS	0800	00A0
SMRADS	0800	00A0
MDRADS	0000	0000
VSXADS	0800	00A0
MEMADS	0800	00A0

Table 13–3. VSC Section Values for Multidrop VSC

ADS	MD"	MI"
VCLADS	0800	00A0
SMRADS	0000	0000
MDRADS	0800	00A0
VSXADS	0800	00A0
MEMADS	0800	00A0

Table 13–4. VSC Section Values for Code Rate Generator Boards

ADS	MD"	MI"
VCLADS	0000	0000
SMRADS	0000	0000
MDRADS	0000	0000
VSXADS	0000	0000
MEMADS	0000	0000

Section 16: VITAL SERIAL OFFSETS AND OFFSET INCREMENTS

A PD-SUM is calculated based on offset and increment values for vsoads, smrads, mdrads, vsxads and vsiads codewords.

Vital Serial Offsets and Offset Increments PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 17: RAM ADDRESS REPORT

RAM and VS RAM buffers not vitally cleared are reported by both reports. A PD-SUM is calculated based on output addresses of outputs which have stored recheck values.

The buffers not vitally cleared that are reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

RAM Address PD-SUM reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 18: MEMORY CONSTRAINTS

A Memory constraint PD-SUM is calculated by the ADV Consolidation function based on the characters in the memory constraint section of the ADV report.

Memory Constraint PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 19: TIMER MINIMUM AND MAXIMUM VALUES

A PD-SUM is calculated based on default min/max timer values.

Timer PD-SUMs reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 20: SOFTWARE SIGNATURE VALUE

Software signature and Site ID values reported in the CAA report ('name'.VCR) and ADV report ('name'.ACR) must be identical.

Section 21: SHADOW BANK MEMORY OFFSET DATA REPORT

Where shadow memory banks are used, the report shows that successive bank offsets increase by four bytes and that the memory from the bank 1 offset up to the offset for each successive bank has been zeroed. The number of shadow banks used depends on the size of the application logic.

For all shadow banks that contain expression results, verify that each shadow bank offset address increases by four (4) bytes from the previous one. Verify that all memory locations from the offset address of the first shadow bank to the offset address of each successive bank are zero (0000).

Section 22: DIGISAFE REPORT SUMMARY

This section only exists if the CAA version supports DigiSAFE communications.

For each DigiSAFE message, a PD-SUM is calculated based on the DigiSAFE Message Input ADS and DigiSAFE Message Output ADS codewords and transitional values used to convert between the parameters NISAL values and the parameters checksum contribution in the DigiSAFE message.

The DigiSAFE Summary PD-SUM reported in the CAA report (‘name’.VCR) and ADV report (‘name’.ACR) must be identical.

A second PD-SUM is calculated by the ADV based on the content of the DigiSAFE Timestamp Threshold (DTTADS) ADS structure.

The DTT ADS Summary PD-Sum reported in the CAA report (‘name’.VCR) and ADV report (‘name’.ACR) must be identical.

The Equipment ID numbers reported by the ADV in this section must be the same as those that were originally entered as compiler inputs and must be unique to each DigiSAFE equipment.

SECTION 14 – ADV COMPARE PROGRAM

The Application Data Verifier Compare (iVPIADVC) Program processes the output from two runs of the Application Data Verifier (ADV), producing a listing that enables the user to determine changes or differences in the application data represented (typically, old and new versions of the same application). THE iVPIADVC PROGRAM DOES NOT DETERMINE THE VALIDITY OF ADV OUTPUTS; the individual output files must still be evaluated for correctness.

iVPIADVC requires the direct, unedited, .LSV ADV output files; for correct operation, these files must be produced using the ADV symbol table option (in CAA versions 025C and later, the ADV always produces symbol table data and no user option is needed).

14.1 PROGRAM OPERATION

Program operation is summarized as follows:

1. Symbol tables are produced by reading the ADV file symbol table sections.
2. The symbol tables are listed in parallel format, or alphabetically sorted and compared depending on the option selected.
3. The Vital input and output sections of the two ADV files are located and compared.
4. The Vital serial input and output message sections are located and compared.
5. The Vital serial pd-sum and checkword sections are located and compared.
6. The Boolean expression sections are located, and arrays of expression results are built for each file.
7. Depending on option selected, the expressions are either listed in pairs by equation number, or compared based on result names.

If the program encounters file errors or incomplete data, an appropriate error message and abort is issued.

14.2 REPORTS

These reports appear in the compare program's output file:

- Symbol Table Report: Symbol table data from both ADV files are listed side-by-side. If the comparison option is selected, old file symbols are sorted alphabetically and an attempt is made to match them with symbols from the new file. Symbols with unmatched names are marked NOT FOUND. Symbols with matching names but different types are marked TYPE.
- Vital Input, Output Reports: Vital input and output board information is listed side-by-side. Differences in board type or address as well as any differences in parameter names are marked.
- Vital Serial Message Reports: Vital serial input and output messages are listed side-by-side. Differences in message length or message parameter name contents are marked.
- Vital Serial Codeword Report: Vital serial message pd-sums and calculated checkwords are listed side-by-side; any differences are marked. Differences found in this area indicate that changes were made in the link and block numbers assigned in the CAA input Link Definition File, or in message length. The CAA input files must be examined to determine the exact changes made. Certain CAA packages also list vital serial links and blocks directly in a separate report; see below.
- Vital Serial Links and Blocks Comparison Report: This report is available in the CAA packages included in CAAPE 007 and later. Vital serial board and VSOE node types, message lengths and assigned link and block numbers are listed side-by-side; any differences are marked. Differences found in this area indicate that vital serial boards and VSOE nodes were rearranged or that the properties of their links were modified.
- DigiSAFE Equipment ID Comparison Report: This report is only available in CAAs supporting DigiSAFE communications. For each DigiSAFE message, DigiSAFE Equipment numbers are listed side-by-side, any differences are marked.
- Boolean Equation Report: Boolean equations are listed in oldfile-newfile pairs. If the comparison option is selected, an attempt is made to match the old file equations based on result names; equations are considered equivalent if they have the same result lists. Unmatched equations from one or the other file are marked NOT FOUND. Equations with matching result lists are compared product term by product term; any differences are marked. Time delay equations with different time delay amounts are marked.

The sets of three time delay equations generated by the CAA and listed in the .LSV files are condensed back into the original single user-entered equation. This eliminates certain CAA-generated data that sometimes caused the ADV Comparison Program to erroneously report equation differences. Later versions of the program also mark equations that compare but are not in the same order differently from equations that are actually different or are not found in both input files.

APPENDIX A – APPLICATION DATA VERIFICATION (ADV) IVPI DATA SHEET

A.1 INTRODUCTION

This section contains the data sheet checklist to record all necessary process steps required to validate information contained in the iVPI application before beginning revenue service.

Retain all data sheets for future reference in the location prescribed by the rules of the local governing authority.

A.2 SAFETY PRECAUTIONS

⚠WARNING

INTENDED SAFE FUNCTIONALITY OF THE IVPI SYSTEM MUST BE VERIFIED

The safety of the application logic as written is the responsibility of an experienced signaling engineer—CAAPE does not make any determination regarding the inherent safety of the logic equations that were entered.

Verifying the accuracy with which CAAPE converted the experienced signaling engineer's application data into PROM data structures is aided by CAAPE, but the signaling engineer must make a final determination using information supplied by CAAPE. CAAPE’s compilers are not themselves Vital programs. An additional independent process is needed to verify that the compile was done correctly. This process is required for all Vital applications.

An experienced signal engineer must verify the safety of the iVPI data and its application. It is the signaling engineer's responsibility to verify the correctness of the iVPI input data in that it accurately represents the intended safe functionality of the iVPI system. Furthermore, "verify the correctness" means that the signaling engineer (1) is required to compare the input and output data files to verify the CAA has operated correctly and (2) must test the iVPI application in its intended environment before it can be placed in revenue service.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

IVPI APPLICATION MUST BE VALIDATION TESTED

Prior to revenue service, validation testing must confirm all iVPI application logic is correct and consistent with application requirements.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

IVPI APPLICATION MUST BE FIELD TESTED

Field testing of a iVPI application is required before placing the location into revenue service. The customer's testing plan and safety plan define the testing requirements for the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

VERIFIER MUST BE DIFFERENT THAN DESIGNER

The experienced signaling engineer responsible for verification (the Checker or Verifier) using the ADV checklist and creating the report shall be independent from the signaling engineer responsible for designing (the Designer) the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

ADV INPUT DATA MUST BE VERIFIED SEPARATELY—PRIOR TO ADV PROCESS

Vital system operation requires that the Boolean equations in the Vital application logic must be written correctly, so that by executing the logic, the iVPI system operates safely in accordance with the rules of the transit or railroad authority.

The Application Data Verifier (ADV) output report provides a means to compare and verify equivalence between the input and the output application data.

However, the Application Data Verifier neither determines the safety suitability of the Boolean expression list nor determines the validity of certain encoded iVPI application data. The input data to the ADV process must be verified for safety separately, prior to the ADV process, and the safety and suitability of the input data is the responsibility of the experienced signaling engineer.

The ADV does, however, issue warnings and error messages as a result of non-vital data checking to alert the experienced signaling engineer to possible discrepancies.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

A.3 IVPI ADV CONSOLIDATION CHECK FORM

iVPI ADV Consolidation Check Form

CONTRACT NAME: (FROM .VCR FILE)			
EQUIPMENT LOCATION: (FROM .VCR FILE)			
VPI PROGRAM NUMBER AND REV: (FROM .VCR FILE)			
VPI PROGRAM NUMBER AND REV: (FROM .ACR FILE)			
ADV RUN DATE: (FROM .ACR FILE)			
ADV RUN TIME: (FROM .ACR FILE)			
VPI CAA NUMBER AND REV: (FROM .ACR FILE)			
COMPILE DATE: (FROM .ACR FILE)		**TIME:**	
CHECKER'S NAME:			
TODAY'S DATE:			
COMMENTS:			

A.4 ADV TROUBLESHOOTING GUIDELINES

If any errors, discrepancies, or unexpected responses are encountered, the Application Data Verification Consolidation Check stops. Any anomalies must be resolved before continuing with the ADV Consolidation Check.

For any unexpected response:

1. Stop the ADV Consolidation Check; do not continue
2. Resolve the issue:
   - Some suggestions to consider:
     - Check for system messages near the end of the .LSV file for possible indications of discrepancies.
     - Check that the application uses all the defined inputs/outputs.
     - Make certain the .VCR and .ACR files are from the same compilation Date and Time.
     - Vital serial link and/or block assignments may need reassignment.
3. Rebuild
4. Start the ADV Check from the beginning
5. Unexpected response in same Verification Section? Contact Alstom Customer Service at 1-800-717-4477.

A.5 ADV PREPARATION

1. Locate the following:

.VCR file	Locate these files in the associated project folder contained in the CAAPE's "Apps" folder. (Use a text editor or file view utility to view / print the files as required for this verification procedure).
.ACR file	
.LSV file	
.LVC file	

2. Print a copy of Appendix A (this section).
3. Print a copy of the iVPI CAAPE .VCR file for this application.
4. Print a copy of the iVPI CAAPE .ACR file for this application.
5. Complete the iVPI ADV Consolidation Check Form in Section A.3.
6. Refer to the .VCR and .ACR file printouts, Header section, and verify the .VCR file Date and Time matches the .ACR file Compile Date and Time.

Do the Dates and Times match exactly?

Response (circle one)	Action
Yes	Continue to Step 7.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

7. View the .LSV report file and scroll to the "**System Message/Error Summary**" section located near the end of the file. If any errors were found, the word "**ERROR**" appears to the left of the message/error. System messages are normal but system errors are not.

Example:

(PMMEM) ERROR CHKMEM PDSUM NOT EQUAL TO DEFAULT KMCHK: KRESLT= 7EF26BAB KMCHK= 09F6033B
(PRYADS) GROUP AND PORT COUNTS CHECK OK
(PVSTSM) ERROR ENCODED TRUE PREK CONT B3E01A1B IS NOT CORRECT.

Are any *errors* reported in the "System Message/Error Summary" of the .LSV file?

Response (circle one)	Action
Yes	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.
No	Continue to SECTION A.6 – ADV Consolidation Check.

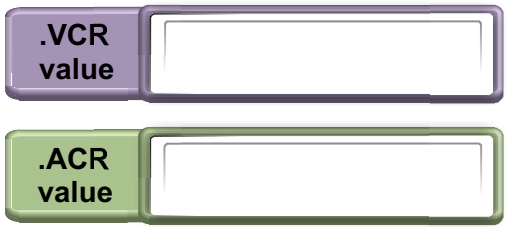
A.6 ADV CONSOLIDATION CHECK

VERIFICATION SECTION 1: SYMBOL TABLE REPORT

No verification required. Continue to Verification Section 2.

VERIFICATION SECTION 2: DUPLICATE NAMES REPORT

Refer to the .VCR and .ACR file printouts, Verification Section 2, and record the 8-digit hexadecimal value for the **DUPLICATE NAMES PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 3.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 3: DUPLICATE ADDRESS REPORT

Referring to the .ACR file printout, Verification Section 3, does it contain the following statement: "**NO DUPLICATE ADDRESSES FOUND**"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 4.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 4: VITAL INPUT REPORT

4a) Refer to the .VCR and .ACR file printouts, Verification Section 4, and fill in the table below with the requested information.

	Compiler Report (from .VCR file)						ADV Report (from .ACR file)					
	CH 1	SGRP ***	BRD ***	CH 2	SGRP ***	BRD ***	CH 1	SGRP ***	BRD ***	CH 2	SGRP ***	BRD ***
*→												
*→												
*→												
*→												
*→												
*→												
*→												
*→												
*→												
*→												
*→												
*→												
*→												

*→ This set (SGRP/BRD) of letters must match: .VCR file to corresponding channel (CH) values from the .ACR file.

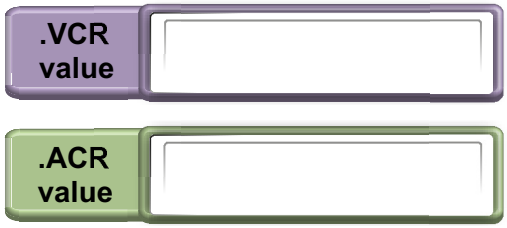
*** This set (SGRP/BRD) of letters must be unique as read down the table.

4b) Verify the values from the .VCR file exactly match the corresponding values from the .ACR file.
and
The Vital Input Board signatures (BRD), as read down the table, are unique for each supergroup (SGRP) signature.

All information verified as correct?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 4c.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

4c) Refer to the .VCR and .ACR file printouts, Verification Section 4, and record the 8-digit hexadecimal value for the **VITAL INPUT PD-SUM** for each.

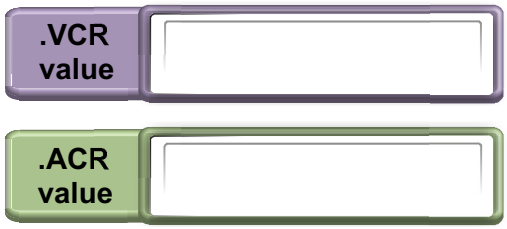


Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 5.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 5: VITAL OUTPUT REPORT

5a) Refer to the .VCR and .ACR file printouts, Verification Section 5, and record the 8-digit hexadecimal value for the **VITAL OUTPUT PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 5b.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

5b) Referring to the .VCR file printout, Verification Section 5, check for a **VITAL OUTPUT BOARDS** table.

If the CAA report ('name'.VCR) contains a Vital Output Boards table, verify each SIG PROM value is unique in the table.

If the CAA Report does not contain a Vital Output Boards table, refer to the .LVC file, Board Report Sections and verify the SIGNATURE PROM for each board is unique.

Are all SIGNATURE PROM values unique?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 6.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 6: VITAL SERIAL COMMUNICATION CONFIGURATION

No verification required. Continue to Verification Section 7.

VERIFICATION SECTION 7: VITAL SERIAL REPORT SUMMARY

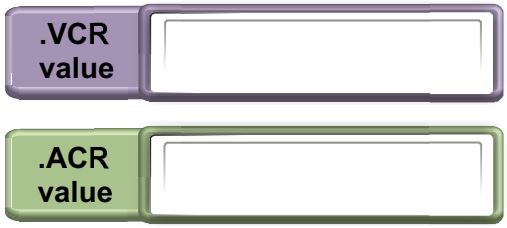
NOTICE

Verification is only applicable if the iVPI system contains Genrakode track processor (**GTP**) **boards**, code rate generator (**CRG**) **boards**, **and/or VSOE2 messages**. If neither of these types of boards or messages are included in the system, the PD-Sum values for these verifications ~~should~~must all be equal to "00000000".

7a) Referring to the .VCR and .ACR file printouts, Verification Section 7, are both values for the **VITAL SERIAL SUMMARY PD-SUM** "00000000"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 8.
No	Continue to Verification Section 7b.

7b) Refer to the .VCR and .ACR file printouts, Verification Section 7, and record the 8-digit hexadecimal value for the **VITAL SERIAL SUMMARY PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 7c.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

7c) Refer to the .ACR file printout, Verification Section 7, and record the **VITAL SERIAL SYSTEM SOFTWARE NUMBER SELECTIONS** (GTP and VSOE/ VSOE2 nodes will not have a Vital Serial System Software Number).

This information is for customer reference only.

CRG 1 Software	
CRG 2 Software	
CRG 3 Software	

Continue to Verification Section 7d.

7d) Refer to the .ACR file printout, Verification Section 7, and record the **VITAL SERIAL LINK AND BLOCK ASSIGNMENTS**.

		Channel 1			Channel 2		
BD	**R/X**	**Link #**	**Block #**	**Block, Range**	**Link #**	**Block #**	**Block, Range**
1	**R**						
1	**X**						
2	**R**						
2	**X**						

		Channel 1			Channel 2		
BD	**R/X**	**Link #**	**Block #**	**Block, Range**	**Link #**	**Block #**	**Block, Range**

		Channel 1			Channel 2		
BD	**R/X**	**Link #**	**Block #**	**Block, Range**	**Link #**	**Block #**	**Block, Range**

		Channel 1			Channel 2		
BD	**R/X**	**Link #**	**Block #**	**Block, Range**	**Link #**	**Block #**	**Block, Range**

		Channel 1			Channel 2		
BD	**R/X**	**Link #**	**Block #**	**Block, Range**	**Link #**	**Block #**	**Block, Range**

7e) For each board and direction of communication (receive(R) or transmit(X)), are: Link numbers and block numbers uniquely assigned to each board (BD) and direction of communication (R / X)?
and
Ranges of blocks numbers do not intersect between boards and direction of communication?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 7f.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

7f) For each board and direction of communication (receive(R) or transmit(X)), do CH 1 and CH 2 have the same information?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 8.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 8: VITAL SERIAL INPUT MESSAGE PARAMETERS

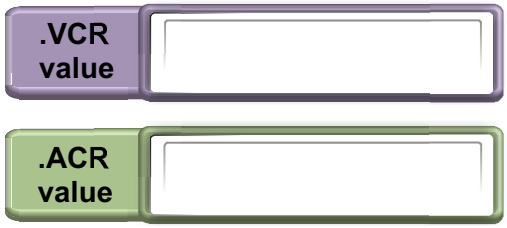
NOTICE

Verification is only applicable if the iVPI system contains Genrakode track processor (**GTP**) **boards**, code rate generator (**CRG**) **boards, and/or VSOE2 messages**. If neither of these types of boards or messages are included in the system, the PD-Sum values for these verifications must all be equal to "00000000".

8a) Referring to the .VCR and .ACR file printouts, Verification Section 8, are both values for the **VITAL SERIAL INPUT PD-SUM** "00000000"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 9.
No	Continue to Verification Section 8b.

8b) Refer to the .VCR and .ACR file printouts, Verification Section 8, and record the 8-digit hexadecimal value for the **VITAL SERIAL INPUT PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 9.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 9: VITAL SERIAL OUTPUT MESSAGE PARAMETERS

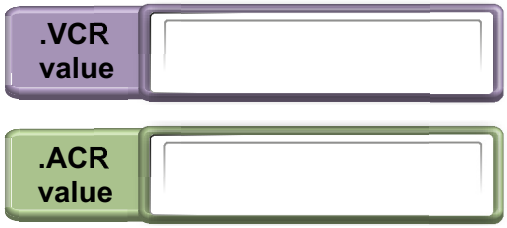
NOTICE

Verification is only applicable if the iVPI system contains Genrakode track processor (**GTP**) **boards**, code rate generator (**CRG**) **boards, and/or VSOE2 messages**. If neither of these types of boards or messages are included in the system, the PD-Sum values for these verifications must all be equal to "00000000".

9a) Referring to the .VCR and .ACR file printouts, Verification Section 9, are both values for the **VITAL SERIAL OUTPUT PD-SUM** "00000000"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 10.
No	Continue to Verification Section 9b.

9b) Refer to the .VCR and .ACR file printouts, Verification Section 9, and record the 8-digit hexadecimal value for the **VITAL SERIAL OUTPUT PD-SUM** for each.



Are the two entered values identical?

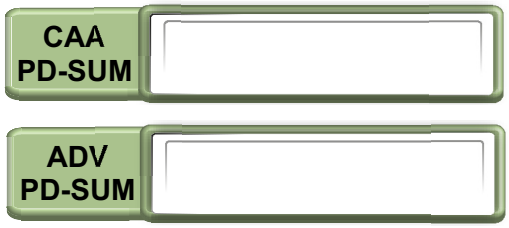
Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 10.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 10: BOOLEAN EXPRESSION REPORT

10a) Referring to the .ACR file printout, Verification Section 10, does it contain the following statement: "**ADV CH1 EQUATIONS MATCH CH2 EQUATIONS**"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 10b.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

10b) Refer to the .ACR file printout, Verification Section 10, and record the 8-digit hexadecimal value for the **CAA PD-SUM** and **ADV PD-SUM**.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 10c.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

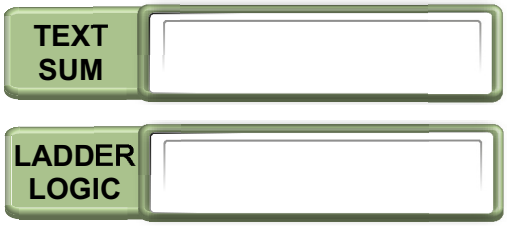
10c) Referring to the .ACR file printout, Verification Section 10, does it contain the following statement: "**NO GRAPHICAL LOGIC DEPENDENCIES**"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 11.
No	Continue to Verification Section 10d.

10d) Referring to the .ACR file printout, Verification Section 10, does it contain the following statement: "**LADDER LOGIC MATCHES LOGIC TEXT**"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 10e.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

10e) Refer to the .ACR file printout, Verification Section 10, and record the 8-digit hexadecimal value for the **TEXT SUM** and **LADDER LOGIC SUM**.

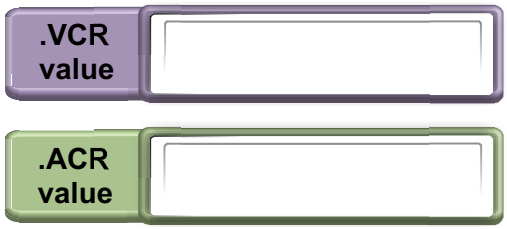


Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 11.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 11: EXPRESSION EVALUATION REPORT

Refer to the .VCR and .ACR file printouts, Verification Section 11, and record the 8-digit hexadecimal value for the **EXPRESSION EVALUATION PD-SUM** for each.

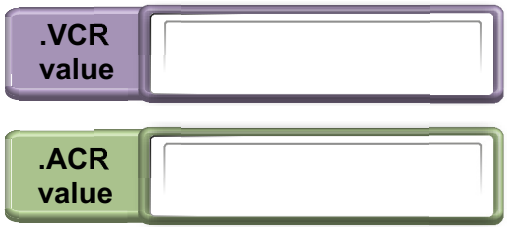


Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 12.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 12: VRD CHECKWORD REPORT

Refer to the .VCR and .ACR file printouts, Verification Section 12, and record the 8-digit hexadecimal value for the **VRD CHECKWORD PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 13.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 13: CHECKWORD SIZE

Refer to the .VCR and .ACR file printouts, Verification Section 13, and record the size value for each Checkword.

Buffer	Compiler Size (from .VCR file)	ADV Size (from .ACR file)
MCKSUM		
CHKIN		
CHKIB		
CHKIA		
CHKIT		
CHKOC		
CHKOTC		
CHKCS		
CHKLA		
CHKCR		
CHKLAT		
CHKX		
CHKY		
CHKYN		
CHKMEM		
CHKTM0		
CKTM12		
CKTM23		
CKTM34		
CHKDUM		

Do the values from the .VCR file exactly match the values from the .ACR file?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 14.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 14: EXPLANATION OF CHECKWORD SIZE

No verification required. Continue to Verification Section 15.

VERIFICATION SECTION 15: DISPLACEMENT AND INCREMENTS REPORTS

15a) VPI CPU/PD[1] DISPLACEMENTS AND INCREMENTS standard values:

ADS	MD	MD"	MI	MI"	RD	RD"	RI	RI"
XMRADS	A000	0800	0400	00A0	4000	0800	0100	0080
WMADS	A000	0800	0400	00A0				
WRADS					4000	0800	0100	0080
MEMADS		0800						

Record the VPI CPU/PD DISPLACEMENTS AND INCREMENTS from the .ACR file.

ADS	MD	MD"	MI	MI"	RD	RD"	RI	RI"
XMRADS								
WMADS								
WRADS								
MEMADS								

1. The Vital board information on the report is displayed under the heading "VPI CPU/PD", regardless of actual board type.

Are all entered values identical to the standard values?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 15b.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

15b) VITAL SERIAL DISPLACEMENTS AND INCREMENTS

No verification required. Continue to Verification Section 16.

VERIFICATION SECTION 16: VITAL SERIAL OFFSETS AND OFFSET INCREMENTS

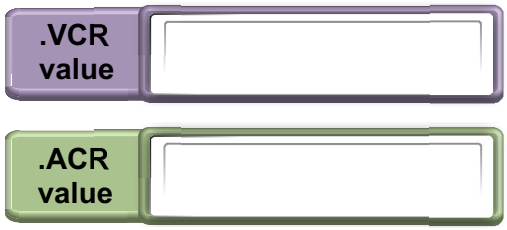
NOTICE

Verification is only applicable if the iVPI system contains Genrakode track processor (**GTP**) **boards**, code rate generator (**CRG**) **boards**, and/or **VSOE2 messages**. If neither of these types of boards or messages are included in the system, the PD-Sum values for these verifications must all be equal to “00000000”.

16a) Referring to the .VCR and .ACR file printouts, Verification Section 16, are both values for the **VITAL SERIAL OFFSETS AND OFFSET INCREMENTS PD-SUM** "00000000"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 17.
No	Continue to Verification Section 16b.

16b) Refer to the .VCR and .ACR file printouts, Verification Section 16, and record the 8-digit hexadecimal value for the **VITAL SERIAL OFFSETS AND OFFSET INCREMENTS PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 17.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 17: RAM ADDRESS REPORT

17a) Refer to the .VCR and .ACR file printouts, Verification Section 17, and fill in the table below with the requested information.

CPU/PD RAM ADDRESSES NOT VITALLY CLEARED:

(The report heading is "CPU/PD", regardless of actual board type)

Buffer Name	Compiler (from .VCR file)		ADV (from .ACR file)			
	Buffer Start	Buffer End	Uncleared Address Start	Uncleared Address End	Buffer Start	Buffer End
	(A)	(B)	(C)	(D)	(A)	(B)
MISC						
OFS1[1]						
TM[2]						
OFS2[4]						
TM`[5]						
TRE				*e*		
W-RCHK						
W-MAIN						*e*
CSITMP[3]				*f*		
V						
UNUSED						
STACK						*f*

1. The OFS1 and OFS2 buffers are only used if CRG boards, GTP boards, VSOE2 communications, or DigiSAFE communications are present in the system.
2. The TM and TM` buffers are only used if vital timers are present in the system.
3. The CSITMP buffer is only used if vital to nonvital (VPI to CSEX) messages are present in the system.

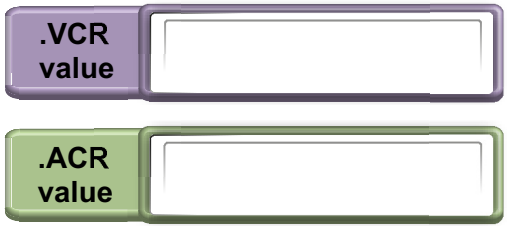
17b) Are the .VCR/.ACR Buffer Start addresses (A)/(A) identical?
and
Are the .VCR/.ACR Buffer End addresses (B)/(B) identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 17c.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

17c) Are the .ACR Buffer Start addresses (A) identical to the .ACR Uncleared Address Start addresses (C)?
and
For Buffer Names: **MISC**, **OFS1**, **TM**, **OFS2**, **TM'**:
Are the .ACR Buffer End addresses (B) identical to the .ACR Uncleared Address End addresses (D)?
and
For Buffer Names: **TRE**, **W-MAIN**, **CSITMP**, **STACK**:
Are the .ACR addresses in the corresponding cell letters (*e* – *f*) identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 17d.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

17d) Refer to the .VCR and .ACR file printouts, Verification Section 17, and record the 8-digit hexadecimal value for the **RECHECK VALUE PD-SUM** for each.

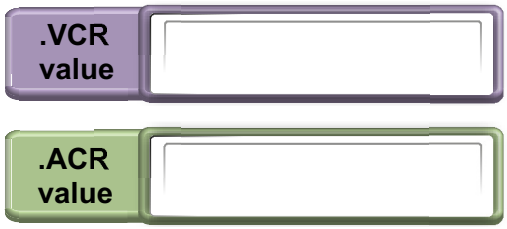


Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 18.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 18: MEMORY CONSTRAINTS

Refer to the .VCR and .ACR file printouts, Verification Section 18, and record the 8-digit hexadecimal value for the **MEMORY CONSTRAINT PD-SUM** for each.

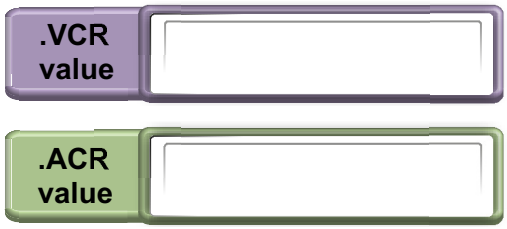


Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 19.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 19: TIMER MINIMUM AND MAXIMUM VALUES

Refer to the .VCR and .ACR file printouts, Verification Section 19, and record the 8-digit hexadecimal value for the **TIMER MIN/MAX PD-SUM** for each.

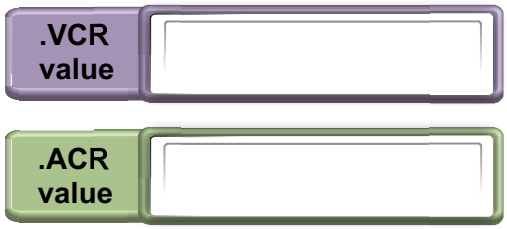


Are the two entered values identical?

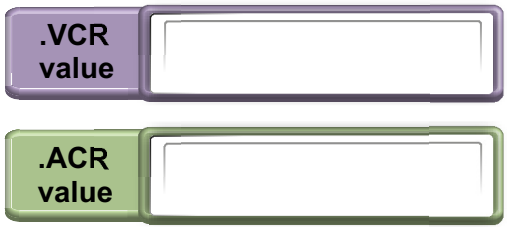
Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 20.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 20: SOFTWARE SIGNATURE VALUE

Refer to the .VCR and .ACR file printouts, Verification Section 20, and record the values for the **SOFTWARE REVISION ID** for each.



Refer to the .VCR and .ACR file printouts, Verification Section 20, and record the values for the **SITE ID** for each.



Are both sets of entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 21.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 21: SHADOW BANK MEMORY OFFSET DATA REPORT

21a) Referring to the .ACR file printout, Verification Section 21, does it contain the phrase: "**SHADOW BANKS 1 TO ___[1] DO NOT CONTAIN EXPRESSION RESULTS**"?
and
Are the .ACR file Offset Addresses for Shadow Banks 1 to ___[1] all equal to "**0000**"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 22.
No	Continue to Verification Section 21b.

1. Represents the number of Shadow Banks listed for this application, with the maximum number being 7.

21b) Application Memory Shadow Banks in use:
Is the .ACR file Offset Address for Shadow Bank 1 **not** equal to zero (0)?
and
Are the .ACR file Offset Addresses for Shadow Banks 2 to ___[1] all equal to "0000"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 22.
No	Continue to Verification Section 21c.

1. Represents the number of Shadow Banks listed for this application, with the maximum number being 7.

21c) For each shadow bank in use, verify that each Shadow Bank Offset Address is always **4-bytes** greater than the previous shadow bank Offset Address, or N/A if zero (0).

NOTICE

All addresses are expressed in hexadecimal format.

Are all responses in the above table circled as either Yes or N/A?

Bank Number	Response (circle one)	Action for all Applicable
2	Yes	Shadow Bank 2 Offset Address is 4 bytes greater than the Shadow Bank 1 Offset Address.
	No	
3	Yes	Shadow Bank 3 Offset Address is 4-bytes greater than the Shadow Bank 2 Offset Address.
	No	
	N/A	
4	Yes	Shadow Bank 4 Offset Address is 4-bytes greater than the Shadow Bank 3 Offset Address.
	No	
	N/A	
5	Yes	Shadow Bank 5 Offset Address is 4-bytes greater than the Shadow Bank 4 Offset Address.
	No	
	N/A	
6	Yes	Shadow Bank 6 Offset Address is 4-bytes greater than the Shadow Bank 5 Offset Address.
	No	
	N/A	
7	Yes	Shadow Bank 7 Offset Address is 4-bytes greater than the Shadow Bank 6 Offset Address.
	No	
	N/A	

Are all responses in the above table circles as either Yes or N/A?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 21d.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

21d) From the .ACR file:
Fill in: SHADOW BANKS _____(A) TO _____(B) DO NOT CONTAIN EXPRESSION RESULTS

Is each Bank in this (A) to (B) range marked N/A in the table above?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 21e.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

21e) From the .ACR file, are the Shadow Banks with Unused Addresses all equal to "**0**"?
Example: SHADOW BANK 2 UNUSED ADDRESSES 039C-03A0 = **0000**

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 22.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

VERIFICATION SECTION 22: DIGISAFE REPORT

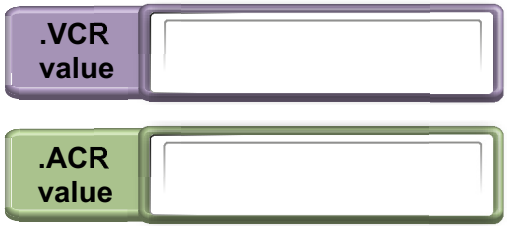
NOTICE

This verification is only for use for iVPI systems that use the VSP vital processor board and system software supporting DigiSAFE communications.

22a) Referring to the .ACR file printout, Verification Section 22, does it contain the following statement: "**NO DIGISAFE MESSAGES FOUND IN SYSTEM**"?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to SECTION A.7 – Conclusion.
No	Continue to Verification Section 22b.

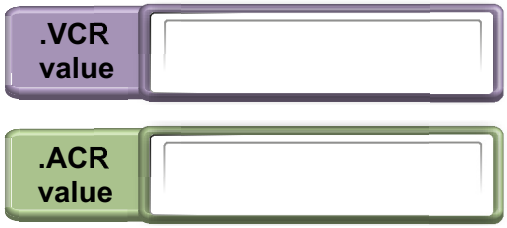
22b) Refer to the .VCR and .ACR file printouts, Verification Section 22, and record the 8-digit hexadecimal value for the **DIGISAFE SUMMARY PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 22c.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

22c) Refer to the .VCR and .ACR file printouts, Verification Section 22, and record the 8-digit hexadecimal value for the **DTT ADS SUMMARY PD-SUM** for each.



Are the two entered values identical?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section 22d.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

22d) Refer to the .ACR file printout, Verification Section 22, and fill in the requested information in the table below.

Zone Controller (ZC)	IN		OUT	
	Source ID (B)	Destination ID (A)	Source ID (A)	Destination ID (B)
1	*g*	*j*		
2	*h*			
3	*i*			

h and *i* exist only if there is more than one (1) zone controller configured in the application.

Are the entries for IN Destination ID (A) and OUT Source ID (A) all the same number?
and
Does the IN Source ID (B) column exactly match the OUT Destination ID (B) column?
and
Are the ID numbers (*g* – *j*) unique for each piece of DigiSAFE equipment?

Response (circle one)	Action
Yes	**SIGN AND DATE:** Verified by: _____ Date: _____ Continue to Verification Section A.7.
No	Do not continue. Refer to SECTION A.4 – ADV Troubleshooting Guidelines for further instructions.

A.7 CONCLUSION

When this **iVPI ADV Data Sheet** has been completely filled out and successfully signed/ verified with no discrepancies, place the following files into the location designated by the governing authority:

- this completed ADV Data Sheet
- .VCR file printout
- .ACR file printout

This is proof that the iVPI Vital Compiler program correctly encoded the application data that will be "burned" into the Vital application PROMs for the VSP boards.

APPENDIX B – ADV COMPARE CHECKLIST

B.1 SAFETY PRECAUTIONS

⚠WARNING

INTENDED SAFE FUNCTIONALITY OF THE IVPI SYSTEM MUST BE VERIFIED

The safety of the application logic as written is the responsibility of an experienced signaling engineer—CAAPE does not make any determination regarding the inherent safety of the logic equations that were entered.

Verifying the accuracy with which CAAPE converted the experienced signaling engineer's application data into PROM data structures is aided by CAAPE, but the signaling engineer must make a final determination using information supplied by CAAPE. CAAPE's compilers are not themselves Vital programs. An additional independent process is needed to verify that the compile was done correctly. This process is required for all Vital applications.

An experienced signal engineer must verify the safety of the iVPI data and its application. It is the signaling engineer's responsibility to verify the correctness of the iVPI input data in that it accurately represents the intended safe functionality of the iVPI system. Furthermore, "verify the correctness" means that the signaling engineer (1) is required to compare the input and output data files to verify the CAA has operated correctly and (2) must test the iVPI application in its intended environment before it can be placed in revenue service.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

IVPI APPLICATION MUST BE VALIDATION TESTED

Prior to revenue service, validation testing must confirm all iVPI application logic is correct and consistent with application requirements.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

⚠WARNING

IVPI APPLICATION MUST BE FIELD TESTED

Field testing of a iVPI application is required before placing the location into revenue service. The customer's testing plan and safety plan define the testing requirements for the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

VERIFIER MUST BE DIFFERENT THAN DESIGNER

The experienced signaling engineer responsible for verification (the Checker or Verifier) using the ADV checklist and creating the report shall be independent from the signaling engineer responsible for designing (the Designer) the iVPI application.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

ADV INPUT DATA MUST BE VERIFIED SEPARATELY—PRIOR TO ADV PROCESS

Vital system operation requires that the Boolean equations in the Vital application logic must be written correctly, so that by executing the logic, the iVPI system operates safely in accordance with the rules of the transit or railroad authority.

The Application Data Verifier (ADV) output report provides a means to compare and verify equivalence between the input and the output application data.

However, the Application Data Verifier neither determines the safety suitability of the Boolean expression list nor determines the validity of certain encoded iVPI application data. The input data to the ADV process must be verified for safety separately, prior to the ADV process, and the safety and suitability of the input data is the responsibility of the experienced signaling engineer.

The ADV does, however, issue warnings and error messages as a result of non-vital data checking to alert the experienced signaling engineer to possible discrepancies.

Failure to comply can degrade the safety performance of the train control system resulting in death or serious injury due to train collision or derailment.

B.2 ADV COMPARE PREPARATION

1. Locate the following:

File Extension	Files Required
.LSV file	ADV Output Files are used by the ADV Compare program. This file is generated by selecting the *.LSV* file (or the iVPI Main project *.VTL* file), ***Actions*** **|** ***Utilities*** **|** ***VPI ADV Compare*** in the CAAPE. When the ADV runs, select item 2 to generate the symbol table in the *.LSV* file.
.LVC file	iVPI CAA Output File
.SYM file	Comparison Program Output File(s)

2. Print a copy of Appendix B (this section).
3. Complete the remainder of the ADV Compare Checklist.

B.3 ADV COMPARE CHECK FORM

Project Location:		ADV Checker: _____ Initials Date
	Old Application	New Application
Application File Name:		
Application File P/N:		
Application Revision		
Application Date:		
CAA Package P/N:	31746-	31746-
CAAPE Package:	31754-	31754-

B.4 ADV TROUBLESHOOTING GUIDELINES

If any errors, discrepancies, or unexpected responses are encountered, the ADV Compare Check stops. Any anomalies must be resolved before continuing with the ADV Compare Check.

For any unexpected response:

1. Stop the ADV Compare Check
2. Resolve the issue:
   - Some suggestions to consider:
     - Check for system messages near the end of the *.LSV* files for possible indications of discrepancies.
     - Check that the application uses all the defined inputs/outputs.
     - Vital serial link and/or block assignments may need reassignment.
3. Run a new ADV Compare
4. Start the ADV Compare Check from the beginning
5. Unexpected response in same Verification Section? Contact Alstom Customer Service at 1-800-717-4477.

B.5 REPORT SECTION

Comparison is performed on new and old VPI/iVPI ADV output files (*.LSV*), using the ADV Compare program. The ADV Compare Program performs comparison automatically and results are noted in the *.SYM* output listing file. The comparison file highlights items that are the same or different between the two *.LSV* files. Identify items checked by (√). Verification of the *.ACR* and *.VCR* files for each compile session that created the *.LSV* files being compared is a prerequisite to this procedure. Differences between files are marked with a pound sign - # - in the first column for quick reference.

B.5.1 SYMBOL TABLE REPORT, .*SYM*

Symbol table data from both ADV files are listed side-by-side. Old file symbols (parameter names) are sorted alphabetically and an attempt is made to match them with parameter names from the new file. Parameters with unmatched names are marked NOT FOUND. Symbols with matching names but different types are marked with #.

	(√) Expected Parameter names

B.5.2 VITAL INPUT REPORT, .*SYM*

Vital input board information is listed side-by-side. Differences in board type or address as well as any differences in parameter names are marked with #.

Compare all input board assignments between old and new *.LSV* files as follows:

	(√) Board slot assignments
	(√) Port assignments
	(√) Cycles of forgiveness assigned to each port

B.5.3 VITAL OUTPUT REPORT, .*SYM*

Vital output board information is listed side-by-side. Differences in board type or address as well as any differences in parameter names are marked with #.

	(√) Board slot assignments
	(√) Port assignments, including flash and current check assignments

B.5.4 VITAL SERIAL INPUT MESSAGE REPORT, ***.SYM***

Vital serial input messages are listed side-by-side. Differences in message length or message parameter name contents are marked with #.

Compare all Vital Serial message parameters between old and new *.LSV* files as follows:

☐ (√) Bit # and name assignments

B.5.5 VITAL SERIAL OUTPUT MESSAGE REPORT, *.SYM*

Vital serial output messages are listed side-by-side. Differences in message length or message parameter name contents are marked with #.

Compare all Vital Serial message parameters between old and new *.LSV* files as follows:

☐ (√) Bit # and name assignments

B.5.6 VITAL SERIAL CODEWORD REPORT, .*SYM*

Vital serial message pd-sums and calculated checkwords are listed side-by-side; any differences are marked. Differences found in this area indicate that changes were made in the link and block numbers assigned in the CAA input Link Definition File, or in message length. The CAA input files must be examined to determine the exact changes made.

☐ (√) Compare all checkwords between old and new *.LSV* files

B.5.7 VITAL SERIAL LINKS AND BLOCKS COMPARISON REPORT, .*SYM*

This report is available in the CAA packages included in CAAPE 007 and later. Vital serial board and VSOE node types, message lengths and assigned link and block numbers are listed side-by-side; any differences are marked. Differences found in this area indicate that vital serial boards and VSOE nodes were rearranged or that the properties of their links were modified.

☐ (√) Compare node types, message lengths, assigned links, and block numbers

B.5.8 DIGISAFE EQUIPMENT ID COMPARISON REPORT, *.SYM*

This report is only available in CAAs supporting DigiSAFE communications. For each DigiSAFE message, DigiSAFE Equipment numbers are listed side-by-side, any differences are marked.

	(√) Compare DigiSAFE Equipment numbers

B.5.9 BOOLEAN EQUATION REPORT, *.SYM*

Boolean equations are listed in oldfile -newfile pairs. If the comparison option is selected, an attempt is made to match the old file equations based on result names; equations are considered equivalent if they have the same result lists. Unmatched equations from one or the other file are marked NOT FOUND. Equations with matching result lists are compared product term by product term; any differences are marked. Time delay equations with different time delay amounts are marked. The program also marks equations that compare but are not in the same order differently from equations that are actually different or are not found in both input files.

	(√) Compare all equations between old and new *.LSV* files

B.6 CONCLUSION

When this **ADV Compare Checklist** has been completely filled out and successfully signed/verified with no discrepancies, place this file into the location designated by the governing authority.

SIGN AND DATE:	
Verified by:	
Date:	

Need help?
Contact Customer Service:

Alstom Signaling Inc.
1025 John Street
West Henrietta, NY 14586
USA

1-800-717-4477
www.alstomsignalingsolutions.com

ALSTOM